

NASA Technical Memorandum 104738

104738
374738
p-263

Making Intelligent Systems Team Players: Case Studies and Design Issues

Volume 1: Human-Computer Interaction Design

Jane T. Malin

Debra L. Schreckenghost

David D. Woods, Scott S. Potter, Leila Johannesen, Matthew Holloway

Kenneth D. Forbus

(NASA-104738) MAKING INTELLIGENT SYSTEMS
TEAM PLAYERS: CASE STUDIES AND DESIGN
ISSUES. VOLUME 1: HUMAN-COMPUTER INTERACTION
DESIGN. (NASA) 263 p

104738

CPCL 12A

uncl is

63/52 0037473

September 1991

NASA

NASA Technical Memorandum 104738

**Making Intelligent Systems Team Players:
Case Studies and Design Issues**

Volume 1: Human-Computer Interaction Design

Jane T. Malin
Lyndon B. Johnson Space Center
Houston, Texas

Debra L. Schreckenghost
The MITRE Corporation
Houston, Texas

David D. Woods, Scott S. Potter, Leila Johannesen, Matthew Holloway
Ohio State University
Columbus, Ohio

Kenneth D. Forbus
Northwestern University
Evanston, Illinois

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

September 1991

Request for Comments

We are interested in making this document useful to the reader. We encourage comments about this document that can help us improve it in future releases.

Please send comments to:

malin@aio.jsc.nasa.gov
schreck@aio.jsc.nasa.gov

Abstract

Initial results are reported from a multi-year, interdisciplinary effort to provide guidance and assistance for designers of intelligent systems and their user interfaces. The objective is to achieve more effective human-computer interaction (HCI) for systems with real-time fault management capabilities (i.e., process monitoring and control during anomalous situations). Intelligent fault management systems within the National Aeronautics and Space Administration (NASA) have been evaluated for insight into the design of systems with complex HCI. Preliminary results include (1) a description of real-time fault management in aerospace domains, (2) recommendations and examples for improving intelligent system design and user interface design, (3) identification of issues requiring further research, and (4) recommendations for a development methodology integrating HCI design into intelligent system design.

Keywords: human-computer interaction, user interface, intelligent system, design guidance, development methodology, real-time fault management

Acknowledgements

We would like to thank all the NASA and contractor personnel who took time to demonstrate and explain their systems to us. Their patience and willingness to share their problems and ideas fostered an ideal environment in which to undertake this work. We also appreciate the time expended by these personnel to review the case study reports for accuracy.

We would like to extend particular thanks to Louis Lollar, David Weeks, and Bryan Walls at NASA Marshall Space Flight Center for their cooperation and encouragement in developing a workshop to present the problems and issues summarized in this paper. We are also grateful to Christine Kelly, Leslie Ambrose, and Kathryn Cooksey from The MITRE Corporation for their review of portions of this report.

Executive Summary

This report documents the initial results from a multi-year, interdisciplinary effort to provide guidance and assistance for designers of intelligent systems and their user interfaces. The objective is to achieve more effective human-computer interaction (HCI) for systems with real-time fault management capabilities (i.e., process monitoring and control during anomalous situations). Intelligent fault management systems within NASA have been evaluated for insight into the design of systems with complex HCI. Fifteen intelligent systems were selected from a variety of aerospace programs, including Space Shuttle, Space Station, unmanned spacecraft, and military and advanced aircraft. Information was collected by interviewing intelligent system developers and users, observing intelligent system demonstrations, and referencing available intelligent system documentation. A review of pertinent literature was also conducted in parallel to the case study.

This report represents work in progress. Preliminary results of this study include (1) a description of real-time fault management in aerospace domains, described in section 3, (2) recommendations and examples for improving intelligent system design and user interface design and issues requiring further research, provided in sections 4 and 5, and (3) recommendations for a development methodology integrating HCI design into intelligent system design, discussed in section 6. These preliminary results will be amplified and formalized in the report *Making Intelligent Systems Team Players: Design Assistance and Guidelines* to be published in Fiscal Year 1993.

The results of the case study are documented in a two volume report. This document is Volume 1, which summarizes the recommendations and issues for HCI design guidance and development methodology. Volume 2 (Malin et al., 1991) provides the detailed analysis of each application in the case study.

Real-Time Fault Management with Intelligent Systems

Fault management has traditionally been performed by teams of human operators (i.e., onboard crew members and ground flight controllers). The introduction of intelligent systems into real-time flight support has altered this traditional mode of operation. The team now consists of human and computer agents. Activities are distributed between these agents. This distribution of activities to software agents and the resulting joint human-computer tasking is a key change in operations. The availability of multiple agents requires allocation of tasks to specific agents and introduces the need for dynamic task allocation for situations where an agent's capabilities to perform a task are altered in real time (e.g., intelligent system fails and the human must take over its task). Issues include managing interruptions in time-critical activities and determining appropriate task assignments. Inappropriate task assignments can adversely affect the operator, complicating performance of tasks and removing opportunities for on-the-job training and development of expertise.

Inherent in the availability of multiple agents is the need to coordinate the activities of these agents, particularly when they are working jointly on a task. These *coordination activities* are independent of and in addition to the fault management activities. When multiple team members are working simultaneously on a fault management task, information from independent activities must be exchanged and related to system behavior. Such a joint effort may be described as *collaboration* between agents of the fault management team. The goal of such collaboration is to allow the operator to understand the intelligent system's point of view by sharing information and providing visibility into intelligent system reasoning. Retrospective explanation (in the form of static rationale and rule trace) is not sufficient for such collaboration and information sharing.

Effective control of the intelligent system also requires that the operator understand intelligent system processing. This includes providing visibility into the intelligent system reasoning and clearly identifying what levels of automation are available, when they are used, and how task are allocated for each level.

Intelligent systems for flight operations support add both enhanced capability and an additional source of error. The operator is responsible to make the final decisions in fault management situations and thus must be able to manage and direct the intelligent system, including the ability to compensate for erroneous behavior. Erroneous behavior can be due to failures of the intelligent software (e.g., inaccurate knowledge base, inadequate performance) or due to erroneous or unavailable input information. Typically, errors in the intelligent system result in shut down of the intelligent system. Error correction is performed off-line, after the operational support period. The intelligent system can be designed, however, to permit on-line error compensation by operator intervention into intelligent system processing. Methods for intervention into the intelligent system reasoning process include (1) altering information used by the intelligent system, (2) redirecting the reasoning process, (3) restarting the intelligent system, and (4) selectively overriding portions of the intelligent system processing. For situations when redirection is ineffectual, the operator can take over all intelligent system task responsibilities.

To retain the capability to manage the monitored process (i.e., the domain process that the team is responsible to manage) and make decisions in failure situations, the operator must maintain an awareness of the overall state and behavior of the monitored process, even when significant portions of the fault management tasks are conducted by the intelligent system. The operator must be able to extract the important information from the barrage of alarms that typically accompany an anomaly and interpret and prioritize these alarms by severity of impact and time-criticality. The operator must also have access to diagnostic information critical to understanding situations affecting the monitored process, including assessment of mission impacts, evaluation of consequences and alternatives, monitoring and execution of procedures, and assessment of functional capability remaining after a failure. Since not all fault management situations can be anticipated, the intelligent system must be designed to support the operator in generating and testing workaround procedures for contingency or unanticipated situations.

A common problem in real-time fault management is overloading the operator with large amounts of dynamic information. The intelligent system should be designed to assist the operator in interpreting and using information, i.e., *managing information*. Managing information overload requires designing the intelligent system and its user interface to assist the operator in interpretation of information. Information context can be used to present information in a more meaningful fashion. Qualitative representation of information can be an effective way to present information consistent with human mental models. Summarization of information and suppression of irrelevant detail are promising techniques for information management, but require further investigation.

Case Study Trends

Several trends in user interface design were found based on the cases studied. One issue that occurred in several systems is a proliferation of windows which can lead to navigation issues concerned with where to find related data. This is complicated by hidden windows and complex menus. This trend points to a lack of workspace design (coordinating the set of views into the monitored process as well as the intelligent system that can be seen together in parallel or in series) and lack of specification of information requirements (what information should the observer be able to extract from observing a particular display -- in isolation and in concert with other displays). With respect to the interface between the human operator and intelligent

system, diagnostic reasoning was typically displayed as chronologically ordered message lists. However, this approach fails to capture the temporal nature of events or distinguish between kinds of messages (e.g., events and actions). The predominant view into the monitored process was physical topology schematics annotated with active data about the state of the monitored process (i.e., color coded digital parameter values or component states). However, there were cases in which this approach did not provide appropriate information for the operator (e.g., did not highlight events or system change).

The net result of these user interface trends is black-box, inscrutable systems where the user interface capabilities inadvertently reinforce barriers between the human operator and the intelligent system as well as between the operator and the monitored process. The lack of transparency in inscrutable systems, caused by lack of needed information or confusing presentation of excessive information, makes it difficult for the operator to visualize the intelligent system's situation assessment and recommendations in relation to the flow of events in the monitored process.

Design Guidance - Recommendations, Examples, and Issues

This report contains numerous recommendations and "Before/After" examples to address identified human-computer interaction design problems and needs. Where further research is needed, a description of the research issue is provided in the design context. Until research helps resolve these issues, designers will make design choices based on engineering judgment and prototyping. Design guidance is provided for both user interface designers and intelligent system designers. It is presented in three broad areas:

- 1) Support for coordination and management of intelligent systems
 - Multi-tasking and dynamic task assignment
 - Collaboration between agents
 - Managing intelligent system errors
- 2) Support for fault management
 - Alarm management
 - Critical diagnostic information
 - Unanticipated situations and workaround
- 3) Support for information management and display
 - Representation and interpretation of information
 - Workspace design

Development Methodology

The development process was initially investigated to identify how to effectively deliver HCI design guidance. Case study observations of the development process, however, identified a more fundamental design problem than delivery of guidance. It was observed during the case study that perceived user interface design problems are often actually intelligent system design problems (e.g., unavailable information, mis-representation of information). These problems result from failing to identify the information that needs to be exchanged between human and computer when performing domain tasks (i.e., information requirements). These information requirements are defined as a part of HCI analysis and design.

Although HCI design can affect the entire support system, including conventional software systems, intelligent systems, and the user interfaces to both, it is usually not considered during system design. At best, the user interface design is influenced by HCI considerations. The user interface primarily addresses presentation *media* (i.e., presentation of information), however, and is not concerned with the information *message* (i.e., information content) that is also a part of HCI design. Since user interface design is frequently done after system design, the potential for HCI design to influence system design does not exist. It is necessary to

modify the design process to include definition of the information requirements for both the support system (both conventional and intelligent software agents) and the user interface.

This process view of system design is often absent during intelligent system design. Designers tend to focus on specific technology aspects of the design (e.g., what software tools will be used) and fail to consider the overall system view of the design (e.g., how will the intelligent system be integrated into the existing support environment). Most types of design guidance address technology issues instead of process issues. A development methodology is needed that addresses design process issues.

Development methodology seems to be the most effective means of integrating HCI design into system design. The methodology should support mapping from a specification of the task into information requirements for both the intelligent system and the user interface. This task description should include both domain tasks and tasks required to coordinate human and computer agents. The methodology should incorporate both information-level and display-level design guidance as an integral part of design. It should provide mechanisms for communication and coordination between members of a multi-disciplinary design team. It should support development of the intelligent system as an integral part of the overall support system, which may include both intelligent and conventional software.

Research Issues

Much work remains to be done to realize the goal of providing effective HCI design guidance for intelligent systems. Team architectures must be defined that specify how fault management tasks are performed, including agent interaction, mode shifts, and dynamic task assignment. This investigation includes identifying what coordination activities are required for shared tasking and information requirements for a controllable, directable intelligent system. Information requirements that support agent communication and collaboration must be identified. These requirements should provide visibility into intelligent system reasoning and permit the operator to understand how and why the intelligent system reached a conclusion, resulting in a shared view of the world between agents. New display designs are needed that accommodate the new types of information and graphic forms and the new mix of tasks and information resulting from the use of intelligent systems for real-time fault management. Finally, an intelligent system development methodology is needed to facilitate use of HCI design guidance in developing information requirements and in designing intelligent systems and their user interfaces. This methodology should support the use of prototyping and storyboards with operational scenarios and task analysis modified to identify agent coordination activities.

Meanwhile, we hope that this document provides significant assistance to designers of intelligent systems for real-time fault management. We also hope that it can be used by the research communities of artificial intelligence, human factors, and software engineering to identify issues to investigate so that even better assistance can be provided to intelligent system designers in the future.

Table of Contents

Section	Page
1.0 Introduction	1-1
1.1 Purpose	1-1
1.2 Objectives and Scope	1-2
1.3 Approach	1-2
1.4 Background on Human-Intelligent System Cooperation	1-3
1.4.1 Question and Answer Dialogs	1-4
1.4.2 Decorating Intelligent Systems with Graphics and Windows	1-6
1.5 Organization of Document	1-10
2.0 Survey of Intelligent Fault Management Systems	2-1
2.1 Approach to the Case Study	2-1
Selection of Cases	2-1
Methods for Collecting Information about Cases	2-1
Analysis of Case Study Information	2-2
2.2 Intelligent Systems in the Case Study	2-3
2.3 Evaluation of the Case Study Process	2-4
3.0 Fault Management Operations with Intelligent Systems	3-1
3.1 Fault Management Task Description	3-1
Monitoring and Fault Detection	3-2
Safing, Mission Impact, and Reconfiguration	3-3
Fault Isolation, Testing, and Recovery	3-4
3.2 Team Resources and Constraints	3-4
3.2.1 Characteristics of Aerospace Operations	3-5
3.2.2 Characteristics of Agents and Teams	3-7
3.2.3 Characteristics of Information	3-11
Source and Authority	3-11
Data Quality	3-12
Timing	3-13
3.3 Agent Activities	3-14
3.3.1 Types of Agent Activities	3-15
Monitoring and Assessment	3-16
Planning and Dynamic Re-planning	3-17
Intervention and Control	3-19
3.3.2 Interaction Between Agents	3-20
Models of Agent Interaction	3-20
Coordination Activities	3-21
3.4 Fault Management Scenarios	3-22
3.5 Fault Management Activities and Information	3-26
3.5.1 Agent Activities for Fault Management	3-27
Monitoring and Assessment	3-27
Planning and Dynamic Re-Plan	3-28
Intervention and Control	3-29
Coordination	3-30
3.5.2 Fault Management Information	3-30
Dynamic Information	3-30
Mission Specific Information	3-31
Baseline Operations Information	3-32
Design Knowledge	3-33

Table of Contents (continued)

Section	Page
4.0 Human-Computer Interaction Design for Intelligent Systems	4-1
Background for the Examples	4-1
4.1 Support for Coordination and Management of Intelligent Systems	4-5
4.1.1 Multi-tasking and Dynamic Task Assignment	4-6
Task Allocation	4-6
Dynamic Task Assignment	4-10
Handover	4-16
Interruption of Operator	4-19
4.1.2 Collaboration between Agents	4-25
Visibility into Intelligent System Activities and Reasoning	4-25
Explanation	4-32
Review	4-41
4.1.3 Managing Intelligent System Errors	4-43
Detection of Intelligent System Errors	4-44
Intervention by Altering Information	4-45
Intervention into Reasoning Process	4-46
Restart of the Intelligent System	4-46
Override of the Intelligent System	4-50
Risks of Intervention in Intelligent System Processing	4-54
4.2 Support for Fault Management	4-55
4.2.1 Alarm Management	4-55
Interpretation of Alarms	4-56
False Alarms	4-60
Redundant Alarms	4-66
4.2.2 Critical Diagnostic Information	4-67
Visibility into Monitored Process	4-67
Evaluation of Consequences	4-68
Predicted Behavior	4-68
Alternative States and Behavior	4-69
Mission Impacts and Procedures	4-71
Functional Capability Assessment	4-73
4.2.3 Unanticipated Situations and Workaround	4-75
4.3 Support for Information Management and Display	4-77
4.3.1 Interpretation of Information	4-77
4.3.1.1 Information Context	4-77
Source of Information	4-77
Quality of Information	4-81
Availability of Information	4-85
4.3.1.2 Qualitative Representation	4-88
4.3.1.3 Summarization	4-90
4.3.2 Workspace Design	4-90
Navigation through Workspace	4-91
Design for Performance	4-96
User Expertise	4-97
Types of User Interfaces	4-99
Lists and Schematics	4-100

Table of Contents (continued)

Section	Page
5.0 Design for Visualization of the Monitored Process.....	5-1
5.1 Introduction.....	5-1
5.1.1 Design for Information Transfer Philosophy	5-1
5.1.2 Levels of Description of a Computer-Based Display System.....	5-2
5.2 Workspace - Proliferation of Windows.....	5-5
5.3 Message Lists and Timeline Displays.....	5-10
5.3.1 Weaknesses of Message Lists	5-14
5.4 Physical Topology Schematics	5-23
5.5 Wading through the Interface Layer.....	5-28
5.5.1 Dissociation of Related Data	5-28
5.5.2 Hidden Functionality.....	5-32
5.5.3 The Problem of Fleeting Data.....	5-34
5.5.4 Overuse of Digital Representations and Underuse of Analog Forms.....	5-34
6.0 The Development Process.....	6-1
6.1 Development Methodology.....	6-1
6.1.1 Task Description.....	6-3
6.1.2 Techniques	6-4
Prototyping.....	6-5
Storyboards.....	6-6
6.1.3 Design Description	6-8
6.2 Design Team.....	6-8
6.2.1 Communication within Design Team	6-8
6.2.2 User Participation in Design	6-10
6.3 Integration into the Support Environment.....	6-11
6.4 Delivery of Design Guidance	6-14
7.0 Summary	7-1
Design Guidance	7-1
Design Methodology.....	7-2
Design Issues	7-3
Appendix A Points of Contact for the Case Study.....	A-1
Appendix B Disturbance Management.....	B-1
B.1 Cascade of Disturbances and Disturbance Management Cognitive Task	B-1
B.1.1 Alarm Handling Trends in the Case Study	B-6
Appendix C Styles of Collaborative Interaction.....	C-1
C.1 Advisory functions.....	C-1
C.1.1 Formulating and Delivering Advice	C-3
C.2 Intelligent Subordinate.....	C-5
C.2.1 Scope of Responsibility	C-7
C.2.2 Control of Attention.....	C-7
C.2.3 Clumsy Automation.....	C-8
C.2.4 New monitoring requirements	C-9
C.3 Cognitive Tools.....	C-9
C.4 Summary: Styles of Human Interaction with Intelligent Systems	C-11

Table of Contents (continued)

Section	Page
Appendix D Description of Fault Management Agent Activities and Information	D-1
D.1 Fault Management Agent Activities	D-1
Monitoring and Assessment.....	D-1
Planning and Dynamic Re-Plan.....	D-2
Intervention and Control.....	D-3
Coordination	D-4
D.2 Fault Management Information	D-5
Dynamic Information	D-5
Mission Specific Information	D-7
Baseline Operations Information	D-8
Design Knowledge.....	D-9
References.....	R-1
Glossary.....	G-1

List of Figures

Figure	Page
1-1	Question and Answer HCI: Interaction Across Barrier to Human Practitioner and Intelligent System Cooperation.....1-5
1-2	Expanded View of Communication Barriers for Dynamic Fault Management Applications1-7
1-3	Interaction Across Barrier Between Human Practitioner and Intelligent System for Dynamic Fault Management.....1-8
1-4	Interaction Across Barriers Where Human Operator Tries to Understand What is Going on in Monitored Process in Relation to Intelligent System's Assessment and Actions1-9
4-1	Illustration of Format Used in the BEFORE and AFTER Examples.....4-4
4-2	BEFORE: Example Illustrating Inappropriate Task Allocation.....4-8
4-3	AFTER: Example Illustrating Improved Task Allocation.....4-9
4-4	BEFORE: Example Illustrating the Loss of Operator Control over the Monitored Process4-11
4-5	AFTER: Example Illustrating the Provision for Operator Control over the Monitored Process4-12
4-6	BEFORE: Example Illustrating Intelligent System with Fixed Planned Activity Sequence4-14
4-7	AFTER: Example Illustrating Intelligent System with Alteration of Planned Activity Sequence4-15
4-8	BEFORE: Example Illustrating Difficulty in Distinguishing Currently Active Mode of Operation4-17
4-9	AFTER: Example Illustrating Clear Distinction of Currently Active Mode of Operation4-18
4-10	BEFORE: Example Illustrating Need to Orient Incoming Operators at Handover4-20
4-11	AFTER: Example Illustrating Intelligent System Orienting an Incoming Operator at Handover4-21
4-12	BEFORE: Example Illustrating No Capability to Handle Interruptions.....4-23
4-13	AFTER: Example Illustrating Assistance in Handling Interruptions4-24
4-14	BEFORE: Example Illustrating No Visibility into Intelligent System's Reasoning4-27
4-15	AFTER: Example Illustrating Visibility into Intelligent Systems Reasoning4-28
4-16	BEFORE: Example Illustrating Inability to Access Intelligent System Reasoning Strategy4-30
4-17	AFTER: Example Illustrating Explicit Representation of Intelligent System Reasoning Strategy4-31
4-18	BEFORE: Example Illustrating No Consideration of Context during Collaboration.....4-33
4-19	AFTER: Example Illustrating Improved Collaboration using Context of On-going Activities4-34
4-20	BEFORE: Example Illustrating Difficulty in Achieving Shared Understanding of Situation between Operator and Intelligent System4-37
4-21	AFTER: Example Illustrating Shared Understanding of Situation between Operator and Intelligent System4-38

List of Figures (continued)

Figure	Page
4-22 BEFORE: Example Illustrating Limited Capability to Review Event History.....	4-42
4-23 AFTER: Example Illustrating Improved Capability to Review Event History.....	4-42
4-24 Example from PDRS HCI Design Concepts Illustrating Alteration of Information Used by Intelligent System.....	4-45
4-25 BEFORE: Example Illustrating Difficulty When the Intelligent System Becomes Disoriented.....	4-47
4-26 AFTER: Example Illustrating Operator Redirecting Disoriented Intelligent System	4-48
4-27 Checkpoint Options Provided by PDRS HCI Design Concepts.....	4-49
4-28 BEFORE: Example Illustrating Intelligent System with an Omission in its Knowledge Base.....	4-52
4-29 AFTER: Example Illustrating the Use of Selective Override to Correct for Knowledge Base Error	4-53
4-30 BEFORE: Example Illustrating Adverse Affects of False Alarms due to Bad Data.....	4-61
4-31 AFTER: Example Illustrating Recovery from False Alarm by Excluding Bad Data from Intelligent System	4-62
4-32 BEFORE: Example Illustrating False Alarms caused by Transient Behavior.....	4-64
4-33 AFTER: Example Illustrating Design that Avoids False Alarms caused by Transient Behavior	4-65
4-34 BEFORE: Example Illustrating Confusion about Source of Information.....	4-79
4-35 AFTER: Example Illustrating Clear Identification of Source of Information.....	4-80
4-36 BEFORE: Example Illustrating Poor Presentation of Alternate Sources of Quality Assessment.....	4-83
4-37 AFTER: Example Illustrating Improved Presentation of Alternate Sources of Quality Assessment.....	4-83
4-38 BEFORE: Example Illustrating Data Mis-interpretation when Fault Management Team is Unaware that Data are Static.....	4-86
4-39 AFTER: Example Illustrating Proper Interpretation of Static Data when LOS is Clearly Indicated on Display	4-87
4-40 BEFORE: Example Illustrating Difficulty in Navigating Among Multiple Windows.....	4-93
4-41 AFTER: Example Illustrating Presentation of Information about What Windows are Available and How to Navigate through Them.....	4-93
4-42 BEFORE: Example Illustrating Operator Getting "Lost" in a Schematic.....	4-94
4-43 AFTER: Example Illustrating Navigation Aid for Moving Through a Schematic.....	4-94
4-44 BEFORE: Example Illustrating Visual Complexity of a Display.....	4-95
4-45 AFTER: Example Illustrating Reduction of Visual Complexity using Overlays to Hide Unnecessary Detail	4-95

List of Figures (continued)

Figure	Page
4-46 BEFORE: Example Illustrating Complex Interaction Sequence using Menus	4-98
4-47 AFTER: Example Illustrating Alternate Method of Interaction using Control Keys	4-98
4-48 Example of Message List.....	4-100
5-1 Levels of Analysis in Computer-Based Display Systems	5-3
5-2 Common Theme in Workspace Design: Default Workspace Augmented with User Configurability	5-6
5-3 Second Approach to Workspace Design: Tiled Windows With User Selecting Which View is Presented	5-8
5-4 Example of Proliferation of Windows and Resulting Clutter Which Impairs Locating Important Information.....	5-9
5-5 Scratchpad Window the Functions Like Clipboard on Apple Macintosh.....	5-11
5-6 Example Illustrating (1) How Information can be Obscured when a Window Overlays the Default View and (2) How Proliferation of Windows Indicates Design Failure.....	5-12
5-7 Example of a Message List Window with Time Stamps, Messages and Format Abstracted from Case Study	5-13
5-8 Example of Sequence of Events That Points Out How Message Lists Obscure Information About Temporal Distance Between Events.....	5-15
5-9 Complete Message List and Partial Message List Containing Same Messages, Some of Which Have "Scrolled Away"	5-17
5-10 Relationships Revealed by Displaying Events on a Timeline	5-18
5-11 Typical Message List.....	5-19
5-12 Messages Shown Categorized by Anomaly Type.....	5-20
5-13 Messages Shown Categorized by Type of Message: Automatic Action, Diagnostic Message, or Recommended Action	5-21
5-14 Messages Categorized by Type into Two Separate Windows: Status and Diagnostic.....	5-22
5-15 Messages in Status Message List "Explained" by Messages in Explanation Window.....	5-24
5-16 Comparing Event-Driven Format to a Plan-Based Format	5-25
5-17 Parameter Value Updates in Schematic Must be Repeated in Messages Window at Right Because They are not Easily Perceived in Schematic	5-27
5-18 Status at a Glance" Display and Paper-Based Physical Topology Schematic from Which It was Derived.....	5-29
5-19 Stages 1-3 in Becoming Informed about an Anomaly	5-30
5-20 Stages 4-7 in Becoming Informed about an Anomaly	5-31
5-21 Wading through the Interface	5-33
6-1 Mapping from Task Description to Information Requirements	6-3
6-2 Example of Requirement Derived from Storyboard (PDRS HCI Design Concepts)	6-7
6-3 Example of HCI Design Guidance, Page 1	6-15
6-4 Example of HCI Design Guidance, Page 2	6-16

List of Figures (continued)

Figure		Page
B-1	Aviation Example of Cascade of Disturbances that can Follow from a Fault.....	B-3
B-2	Model of Anomaly-driven Information Processing in Disturbance Management.....	B-4
C-1	Styles of Human-Intelligent System Interaction: Intelligent System as Advisor and Human as Problem Holder	C-2
C-2	Styles of Human-Intelligent System Interaction: Intelligent System as Subordinate and Human as Supervisory Controller	C-6
C-3	Styles of Human-Intelligent System Interaction: Intelligent System as Cognitive Tool for Human Problem Solver.....	C-10

List of Tables

Table	Page
3-1	Characteristics of Aerospace Fault Management.....3-7
3-2	Examples of Software Modes Observed during the Case Study.....3-10
3-3	Types of Activities Associated with Fault Management Actions.....3-15
3-4	Examples of Methods for Assessing Situation and System Behavior.....3-17
3-5	Examples of Typical Heuristics used in Flight Operations3-19
3-6	Baseline Fault Scenario.....3-23
3-7	Examples where Scheduled Procedures are not Performed3-26
3-8	Examples where Unscheduled Procedures are Performed3-26
3-9	Agent Activities Supporting Monitoring and Assessment3-27
3-10	Agent Activities Supporting Planning and Dynamic Re-Plan3-28
3-11	Agent Activities Supporting Intervention and Control3-29
3-12	Agent Activities Supporting Coordination.....3-30
3-13	Types of Dynamic Information3-31
3-14	Types of Mission Specific Information.....3-32
3-15	Types of Baseline Operations Information3-32
3-16	Types of Design Knowledge.....3-33
4-1	Symbols Used in the Examples.....4-5
4-2	Example of Coding for Message Priority.....4-25
4-3	Queries Promoting Agent Collaboration.....4-40
4-4	Example of Procedure Execution Modes from the OMA Prototypes.....4-51
4-5	Alarm Definition for Space Shuttle Caution and Warning.....4-56
4-6	Alarm Classes for Space Shuttle Caution and Warning.....4-58
4-7	Levels of Severity for Space Shuttle Vehicle Control Anomalies.....4-74

Section 1

Introduction

Flight operations support is characterized by the need to process large quantities of information in high risk situations under real-time constraints. The demands of the aerospace environment on humans can be alleviated by enhanced support software. Assistance in the form of intelligent support software can result in an effective joint cooperative problem-solving capability. In such a joint effort, the *operator* (i.e., any onboard crew member or ground flight controller that interfaces with the intelligent system) and the intelligent system can be considered as a team. The human is the team leader of the cooperative effort and the intelligent system assists the human. The computer provides assistance according to a pre-defined set of capabilities and range of activities.

In 1990, a study was initiated to provide guidance in designing intelligent systems that are effective team members in flight operations support. This study was conducted as part of a Research and Technology Operating Plan (RTOP) for the Artificial Intelligence Division of the Office of Aeronautics, Explorations, and Technology (OAET). This report documents a case study performed in three parts this year as part of the RTOP to investigate design guidance. One part was conducted at the Johnson Space Center (JSC) by Jane Malin of the Intelligent System Branch at JSC and Debra Schreckenghost of the MITRE Corporation. A second part conducted by David Woods of Ohio State University included intelligent systems at other National Aeronautics and Space Administration (NASA) centers. The third part was performed by Kenneth Forbus at Northwestern University and addressed qualitative model-based intelligent systems.

This document represents work in progress. It is intended to assist designers and to highlight issues for researchers. The results are based on a case study and a review of pertinent literature. Study to amplify and formalize preliminary recommendations is planned to continue through 1993, culminating in an updated report on human-computer interaction (HCI) design guidance titled *Making Intelligent Systems Team Players: Design Assistance and Guidelines*.

1.1 Purpose

The purpose of this project is to identify and address HCI issues that are critical to intelligent systems. These issues have been termed "advanced HCI issues" and affect both HCI and intelligent system design. These issues are not unique to intelligent systems, for they impact any system involving multiple, intelligent agents (i.e., human or computer entities that exhibit intelligent behavior in the accomplishment of task goals) working together as a team to accomplish tasks. Intelligent systems are, however, the most commonly occurring systems with these characteristics. This study assumes an expert user, a sophisticated intelligent system, and a complex domain requiring real-time support. The resulting HCI issues are related to the coordination of team activities, multi-tasking in distributed environments, dynamic task allocation between agents, and the management of large quantities of information in real time. For real-time intelligent systems, these advanced HCI issues have turned out to be the central focus of the HCI problem. But such issues are not addressed by traditional computer-human interaction and human factors.

There are two thrusts to this investigation: identifying effective forms of HCI design guidance and integrating this guidance into the intelligent system design process. The source of observational information for this project was the case study of intelligent systems. This report documents the results of that case study.

1.2 Objectives and Scope

The objective of the case study was to identify preliminary guidance in the design of effective HCI with intelligent fault management systems in aerospace applications. HCI design represents a mapping from the tasks of the domain into the mechanics of the visual display. Much of the current work in human factors and experimental psychology focuses on finding basic tenets for HCI design that are independent of the domain of application (e.g., the way that humans scan a visual display). This approach strives to avoid consideration of the task-specific aspects of the problem. Thus most of the existing guidelines are targeted at the display specification of design (e.g., Smith and Mosier, 1986; JSC, May, 1988) and do not address the other aspects of HCI design.

An important stage in the design of effective HCI is the identification of agent activities and information required to perform tasks. Task analysis techniques developed for conventional software assist in identifying the fault management activities performed by agents of the system. Systems with multiple, interacting agents, however, exhibit another class of activities that are not identified by conventional task analysis. These are the activities required to coordinate tasks shared by the human and computer agents. These coordination activities are solely for the purpose of effective interaction between agents and are independent of fault management activities. An important element in accomplishing these activities is access to and management of information from all elements within system (i.e., the monitored process, the intelligent support system, and the human operator). Thus, the emphasis of this study is on assisting designers in identifying and translating fault management tasks into *agent activities* (i.e., actions that have been assigned to specific agents) and *information requirements* (i.e., the information exchanged between the operator and the intelligent system). This specification of agent activities and information requirements will affect the design of both the intelligent system and its HCI.

The results of the case study should be of interest to intelligent system designers and to researchers in the areas of HCI and human factors, artificial intelligence, and software engineering. This document identifies preliminary design recommendations and issues (sections 4 and 5) and discusses the design process (section 6) for intelligent system and HCI design. These results are provided to address the current need for HCI guidance in the development of intelligent system without waiting for more formal guidelines to be available. This document should be used with an awareness that results are preliminary and will be superseded by later documents.

1.3 Approach

A case study of intelligent systems within NASA was performed to identify preliminary design guidance. A number of constraints were used to assist in selecting cases. The investigation has been limited to fault management systems within aerospace domains. For this study, a fault management system is a software support system that assists a flight controller in monitoring real-time fault detection, isolation, and recovery (FDIR). The study was also constrained to human-centered applications, where responsibility for the control of both the monitored process and the intelligent support system lies with the operator. Thus the computer becomes a technical assistant to the operator. This assumption affects the roles that both the human and computer can fulfill.

Information about the applications selected for this case study was acquired by interviewing the developers and users of NASA fault management intelligent systems. These interviews were supplemented by relevant documentation and demonstrations of these applications. The information collected included a domain description, intelligent system functionality, and

supporting HCI capabilities. The design techniques and methodologies used were also documented in an attempt to characterize the design process. Information collected during the case study has been supplemented by extensive literature search and relevant conference reports in the areas of intelligent fault management and human-computer interaction.

Information collected during the case study has been used to identify candidate guidelines, interesting examples, key design issues, promising research areas, effective design methodologies, and characteristics of the design process. The aerospace domain has been evaluated to identify the design constraints and requirements imposed by this environment on the intelligent support system. Fault management activities have been characterized and the information requirements to perform these activities in conjunction with an intelligent system have been discussed. These requirements affect the design of both the intelligent system and its HCI. Guidance in identifying agent activities and information requirements has been provided in the form of recommendations and issues. Examples illustrating recommendations are also provided. These examples reference a hypothetical intelligent system developed for a space-based Remote Manipulator System (RMS). They are based on observations made during the case study and illustrate HCI design before and after use of the recommendation. As the first step in integrating HCI design guidance into the overall intelligent system design process, the design process used to develop intelligent systems in the case study has been characterized and a preliminary format for delivery of guidance has been developed.

The authors also had the opportunity to participate in the design of the HCI for the Space Shuttle Payload Deployment and Retrieval System (PDRS) Decision Support System (DESSY) prototype. HCI design concepts were provided in the form of a paper storyboard. These concepts were structured to emphasize information requirements for PDRS flight support using this intelligent system. This in-depth investigation allowed first hand observation and participation in the design process, permitted testing of techniques for specifying a display, and provided a series of examples that have been incorporated into this report.

1.4 Background on Human-Intelligent System Cooperation

While many of the initial expert systems developed were called consultant systems, these intelligent systems possessed minimum capabilities for supporting cooperative interaction with human practitioners. The interaction was based on a question and answer dialogue; there were minimal capabilities for explanations of machine solutions; the human domain practitioner had few if any means available to inspect the intelligent system's reasoning or control the system as a resource in his or her own problem solving process.

However, it was quickly realized, especially for applications involving supervisory control of dynamic processes, that a more meaningful interaction between human and intelligent machine was required. As a result of this, as well as trends in the spread of interface technology, intelligent systems have been developed that go beyond a question and answer dialogue. These new systems incorporate graphic displays, windowed workstations, and more extensive explanation capabilities, ostensibly to support better interaction.

The need to integrate human and machine problem solvers into an effective cooperative system -- to make artificial intelligence (AI) systems team players -- has been recognized, especially for dynamic fault management applications. To do this requires serious consideration of the coupling between human and intelligent system, as well as the requisite interface capabilities, as an integral part of the design of intelligent systems.

One can think about the problem of human interaction with intelligent systems at two levels. At a concrete level, human interaction can be thought of in terms of the computer interface

between the human operator and the intelligent system -- the graphic displays available, the window structure, the dialog mechanisms that support moving around in the interface or that support communication with the intelligent system. But at a deeper level, design of the interaction between practitioner and the intelligent system requires an explicit definition of the roles of each, as well as consideration of an appropriate cooperative problem solving approach for a particular application. This section explores some of the issues in cooperative problem solving (cf., also, Woods, 1986; Woods, Roth and Bennett, 1990; Robertson, Zachary and Black, 1990 or the bibliography in Woods, Johannesen and Potter, 1990).

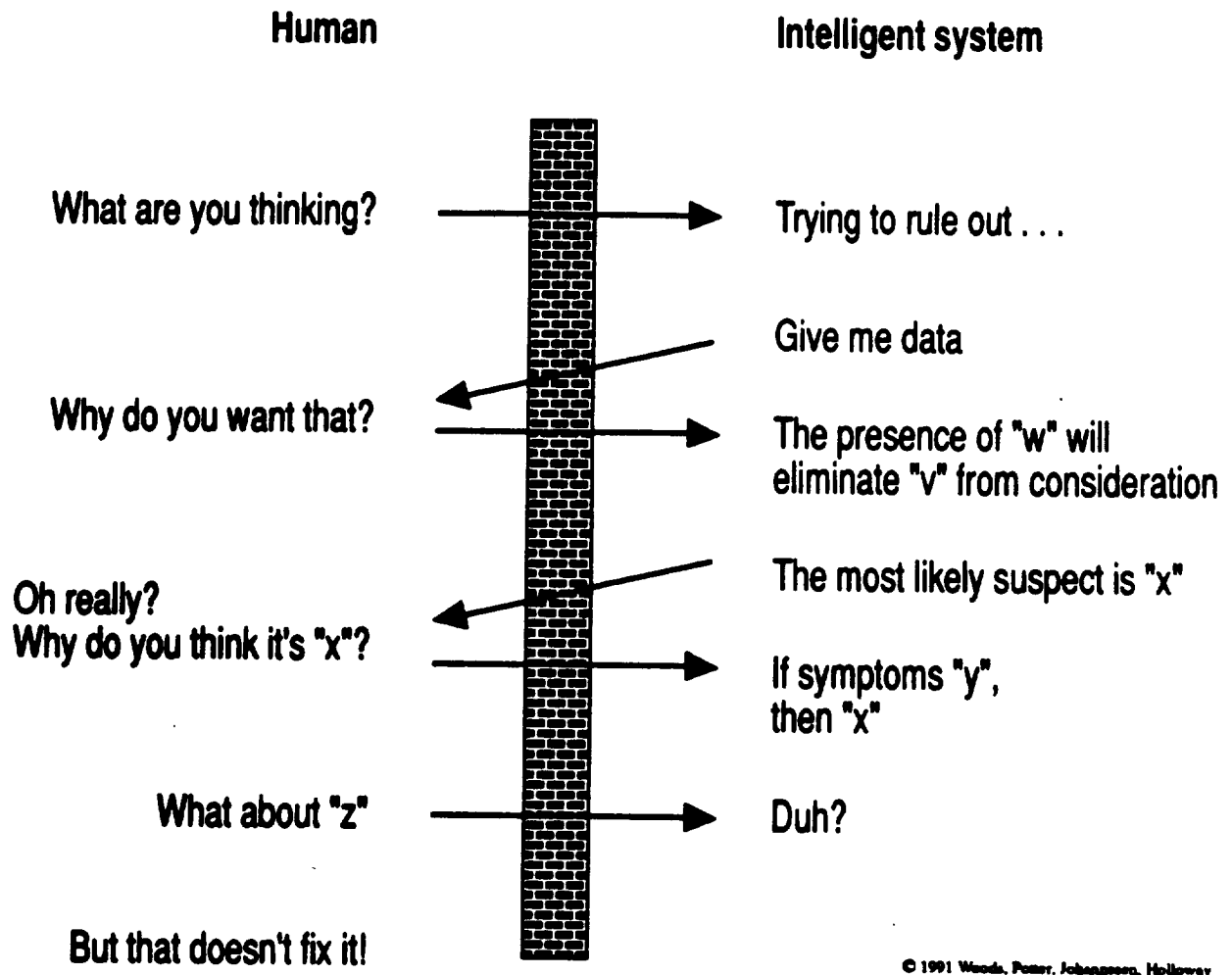
1.4.1 Question and Answer Dialogs

In the beginning, intelligent systems interacted with domain practitioners via question and answer dialogs. The human team member went out and gathered data for the expert system because the expert system was not connected to a database about the state of the device or the monitored process. Similarly, the intelligent system possessed no effector mechanisms and so relied on the domain practitioner to carry out its conclusions about the nature of the fault and how to correct it. However, it was recognized that these systems were not capable of solving all possible problems that might occur; therefore they were called "computer consultants" implying that the human could and should overrule the machine expert whenever he or she determined that it was in error.

Woods has called this style of human-intelligent system interaction the machine-as-prosthesis paradigm (Woods, 1986). In this paradigm, the human's role is to serve as the eyes and hands of the machine. Roth Bennett and Woods (1987) conducted a study of the interaction between technicians and an intelligent system designed in the machine-as-prosthesis paradigm for a troubleshooting application. They found that, even in static troubleshooting situations where there are not any of the dynamism, safing requirements, and mission impact requirements that occur in aerospace contexts, effective performance required the human to play an active role as a full partner or as a supervisor especially in more complicated situations. The machine-as-prosthesis intelligent system failed to provide any interface mechanisms that supported the technicians as partners in the problem solving process; in other words, the AI system failed to be a team player.

The intelligent system in this study used a question and answer dialog as the only means of communication between the human and the intelligent system. The results showed that this form of interaction resulted in a wall between the human and intelligent system (shown in figure 1-1). The active, successful technicians tried to break through this wall to discover the expert system's reasoning process and to manipulate the machine as a resource to help them solve their problem (the human was the problem holder). Thus, the design task in making AI systems team players can be thought of as breaking down that barrier to collaboration or enhancing collaboration by creating effective windows to see through the wall.

The realization that more effective support for collaboration was needed drove people working on the AI research agenda to formulate the problem as -- the machine expert needs to be more intelligent, in the sense that the intelligent system needs to be a better conversationalist. Hence, a relatively large amount of the effort on human-intelligent system cooperation from an AI point of view has been directed at enhancing the machine's natural language capability and enhancing the machine's ability to talk about its own reasoning (cf., the structured bibliography of work on human-intelligent system interaction, Woods, Johannesen, and Potter, 1990).



© 1991 Woods, Potter, Johannsson, Holloway

Figure 1-1. Question and Answer HCI: Interaction Across Barrier to Human Practitioner and Intelligent System Cooperation

It is clear that conversational style interactions as the primary form of communication between the human and intelligent system is inadequate for dynamic fault management applications. One characteristic of these types of situations is that communication demands, information processing demands, decision demands all tend to go up as the severity of challenges to process integrity go up. The communication bandwidth of a conversational dialogue is too low to support timely and effective exchange of information between cooperating agents under these circumstances (cf. e.g., the results on clumsy automation -- Wiener, 1989; Cook et al., 1990).

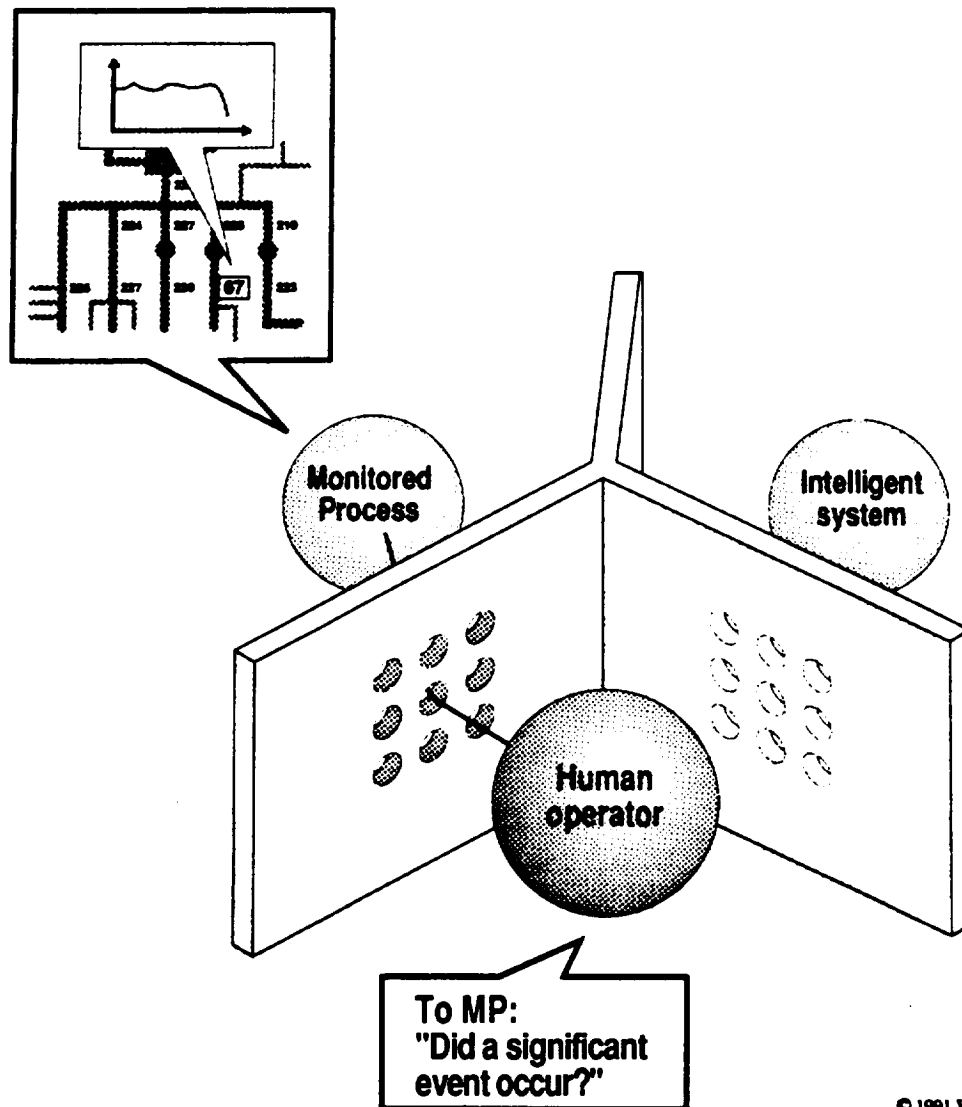
1.4.2 Decorating Intelligent Systems with Graphics and Windows

However, aerospace and many other applications are dynamic real time systems where databases about the state of the monitored process are available. All of the intelligent system cases examined in this project recognized that question and answer style dialogs are inadequate to support the communication requirements between operator and intelligent system for real time systems. And every one of the NASA systems examined included color graphic, multiple window interface capabilities to hopefully expand the communication bandwidth. The question is how to use these technologies to establish collaboration and communication between the human practitioner and the intelligent system.

For aerospace applications, it is necessary to expand on the model in figure 1-1 because the goal of the human operator and the intelligent system is to control/manage an engineered system -- what we will call the monitored process. Figure 1-2 shows a wall between the human and the monitored process. This territory can be addressed in terms of human-computer interaction without any role for intelligent advisory systems. However, the portion of the human-computer interface that is most relevant is concerned with what Woods (1991) calls design for information extraction in HCI -- in other words, how does the computer interface help the human operator understand what is going on in the monitored process.

The introduction of AI systems added a new player to the picture (at least in the typical fashion, although see Appendix C for discussion of cognitive tools). This creates new coordination tasks as the human has to understand what his partner, the intelligent system, is thinking or doing (see figures 1-2, 1-3, and 1-4). But it is important to keep in mind that the ultimate purpose of the human operator in fault management (and the intelligent system) is to track what is going on in the monitored process, recognizing and correcting anomalies.

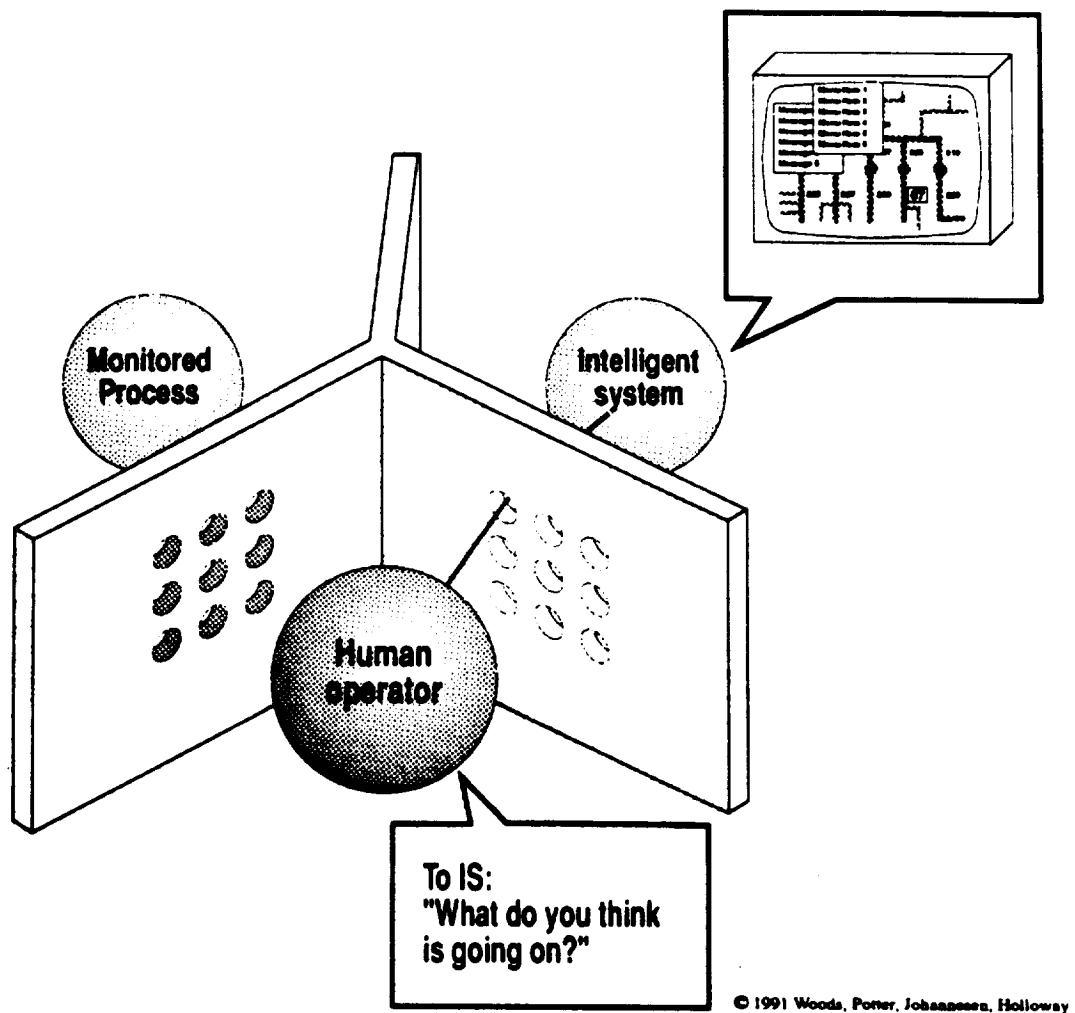
Figure 1-2 shows a situation where an event, a change, has occurred in the monitored process. The window available to the human operator (the representation of the monitored process; cf., Woods, 1991) is a kind of display which was typical of the systems examined in the case study research -- a schematic of the physical topography of some part of the monitored process where the active state is represented by digital values. The user's cognitive task is recognizing if a significant event occurred. The representation of the monitored process affects the ability of the operator to do this.



© 1991 Woods, Potter, Johannessen, Holloway

In this case the human operator is trying to recognize and track significant changes in the monitored process.

Figure 1-2. Expanded View of Communication Barriers for Dynamic Fault Management Applications



In this case the human operator is trying to collaborate with the intelligent system in situation assessment.

Figure 1-3. Interaction Across Barrier Between Human Practitioner and Intelligent System for Dynamic Fault Management

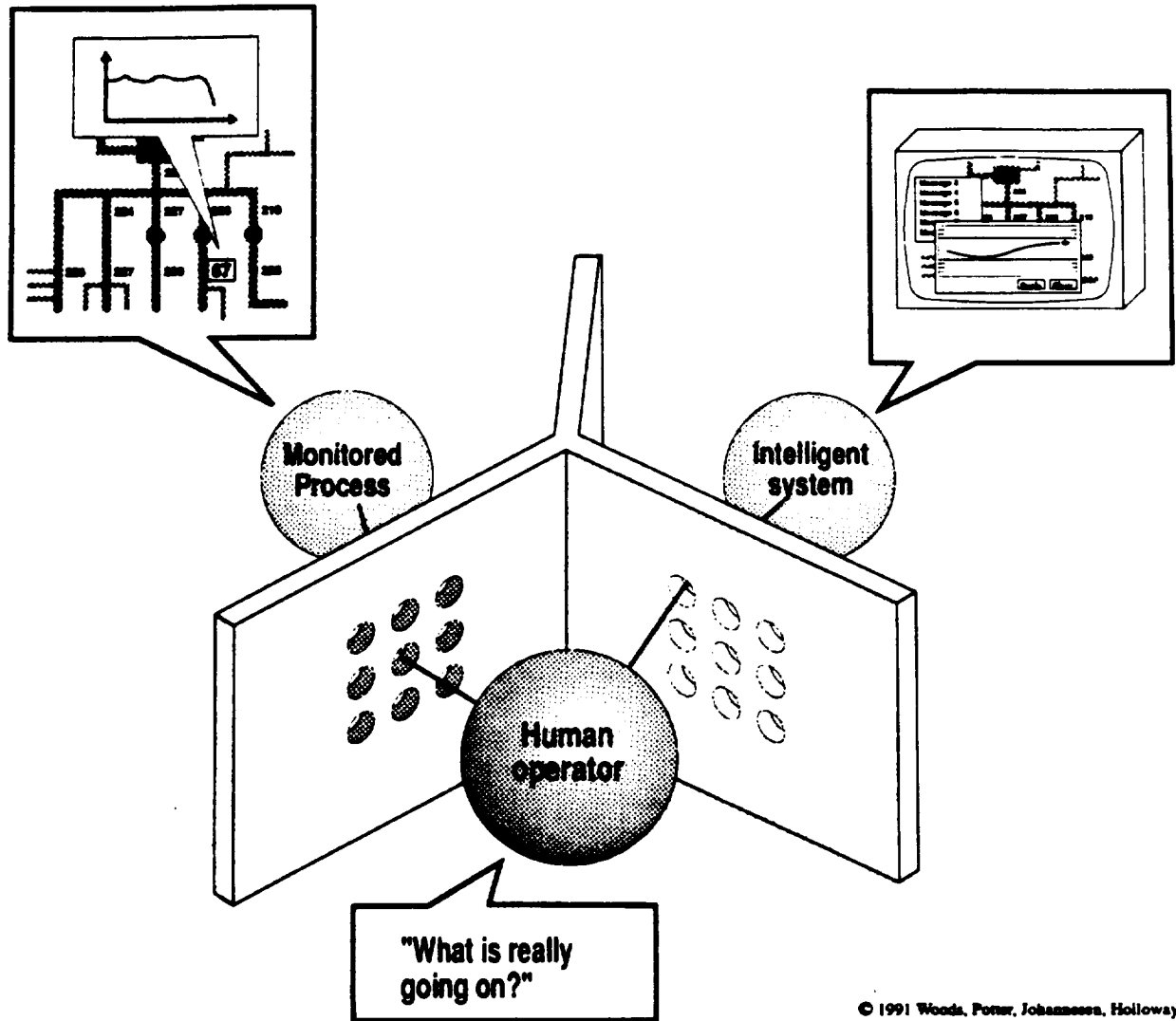


Figure 1-4. Interaction Across Barriers Where Human Operator Tries to Understand What is Going on in Monitored Process in Relation to Intelligent System's Assessment and Actions

Figure 1-3 illustrates the collaboration between the human and intelligent system for this hypothetical scenario. The collaboration is influenced by the kinds of representational windows available to assess the intelligent system's view of the situation. Typically in the systems examined in the case study, there is some kind of message list or menu of options for the kinds of messages or other displays that can be called up. The human must interpret these displays, decide on where additional information may be available within the computer workspace, remember how to get there, examine and integrate the data in order to understand the intelligent system's situation assessment, what faults are present, what corrective actions might need to be taken, what safing actions need to be taken as well as trying to understand how the state of the monitored process is changing (see figure 1-4). Again, the representation of the intelligent system available to the operator affects his or her ability to collaborate with the intelligent system.

The example explored in figures 1-2 through 1-4 shows how designers can inadvertently reinforce barriers that make it difficult to see what the intelligent system is thinking about or what the intelligent system thinks is going on in the monitored process or to see the flow of events in the monitored process. This brings us to one of the major themes that re-occurs throughout the examples, problems, principles and techniques described in this report -- visualization. A very simple criteria to use when evaluating a system is to ask, can I see what is going on? Can someone like the user see what is going on?

This report explores some of the ways that designers inadvertently reinforce barriers resulting in inscrutable systems and identifies some design techniques that break down these barriers enhancing the human operator's ability to visualize the monitored process and the intelligent system.

1.5 Organization of Document

This report is the first volume of a two volume set documenting the results of the case study. This volume (*Volume 1. Human-Computer Interaction Design*) presents preliminary HCI design guidance for intelligent system developers. *Volume 2. Fault Management System Cases* (Malin et al., 1991) contains the interview reports describing each application in the case study.

Section 1 of Volume 1 identifies the problems associated with the design of intelligent fault management systems and describes the resulting objectives of the case study. Section 2 presents an evaluation of the methodology used in the case study. Section 3 describes fault management tasks in aerospace domains and summarizes the information and agent activities required to perform these tasks. Section 4 presents the preliminary recommendations and design issues identified by the case study. Section 5 discusses designing for visualization of the monitored process and intelligent system activity. Section 6 summarizes recommendations and issues related to the design process observed during the case study. Section 7 summarizes the conclusions of the case study. Four appendices are also provided. Appendix A lists the points of contact made during the case study. Appendix B discusses the cognitive aspects of managing disturbances and introduces a model for disturbance management. Appendix C presents the styles of collaborative interaction between human and intelligent software agents. Appendix D summarizes the information required to perform real-time fault management.

Section 2

Survey of Intelligent Fault Management Systems

2.1 Approach to the Case Study

A case study was initiated as a means of collecting information that would assist in the design of human-computer interaction with intelligent systems. The objectives of this case study were to:

- Describe real-time monitoring and fault management in aerospace operations using intelligent system
- Provide preliminary HCI design guidance in the form of recommendations and examples for designers
- Identify HCI design issues for further investigation
- Evaluate the observed design processes for ideas concerning effective delivery of guidance.

The remainder of this section describes the approach used to conduct the case study.

Selection of Cases

The applications selected for this case study were chosen as representative of typical NASA intelligent system design efforts for monitoring and fault management tasks. Prototypes for both Space Shuttle and Space Station were selected. Other applications were selected that support unmanned spacecraft at Jet Propulsion Lab, the X-29 aircraft at Dryden Flight Research Facility, and military aircraft such as the F-15 and F-16 at Edwards Air Force Base. The aircraft applications were chosen because they use the Real Time Data System (RTDS), a hardware and software environment from JSC that provides data acquisition software and intelligent system prototyping tools. See section 2.2 for a description of each application in the case study.

A challenging issue in the selection of cases was determining what traits distinguish an intelligent system from more conventional software. The goal was to select applications that have addressed the advanced HCI issues commonly encountered with intelligent system technology (see section 1.4 for a discussion of these issues). A number of criteria were applied, including the use of an heuristic or model-based design approach and the application of advanced methodologies and concepts.

Methods for Collecting Information about Cases

The report is based on a review of existing research on human-intelligent computer cooperation (Woods, Roth and Bennett, 1990; Woods, Johannesen and Potter, 1990) and on the results of a case study of NASA intelligent fault management systems. The case study examined human interaction styles, human-computer interface capabilities, and the design process used in ongoing NASA projects to develop intelligent fault management systems.

The primary method for collecting information about these applications was interview of the system designer or user. All individuals that were interviewed were very cooperative and information was frequently volunteered. In many cases, these interviews were accompanied by a detailed demonstration of the application. Where possible, available documentation (e.g., briefings, requirements documents, papers, etc.) was used to supplement the information

gained in the interview. Phone calls and additional visits were used to clarify uncertain observations and correct inaccuracies. In one case, the Space Station Module/Power Management and Distribution (SSM/PMAD), a second visit included review of updates suggested on the first visit that had been incorporated into the application, allowing "before and after" evaluation. The final interview reports were reviewed by the individuals that had provided the information to ensure that the application was accurately described. See appendix A for a list of contacts made during the interviews.

For one case -- the Space Shuttle Payload Deployment and Retrieval System (PDRS) Decision Support System -- the authors had the opportunity to participate in the design of the HCI for the intelligent system. For this system, an intelligent prototype existed that had no user interface design. The system developers were flight controllers from the PDRS flight support group. The authors provided HCI consultancy and design concepts to the flight controllers. Information about the system was collected by (1) studying crew and flight controller training manuals and related fault management documentation, (2) interviewing the prototype developer, (3) close review of the prototype, including source code, (4) working meetings with PDRS flight controllers involved in this prototype. The background of one of the authors (Debra Schreckenghost) as a Space Shuttle flight controller was also of material use in this effort. Based on this information, a storyboard of HCI design concepts was developed for the PDRS Decision Support System (Schreckenghost, 1990) using the MacDraw® graphics software. Where possible, examples and suggestions from the case study were incorporated. This storyboard was presented to flight controllers, review comments were elicited, and upgrades were suggested. The storyboard was later implemented by the HCI Lab (Space and Life Sciences Directorate/SP34 at JSC) as an interactive display using the prototyping software tool Prototyper on a Macintosh® and used for demonstration. This effort is part of the ongoing collaboration between the developers of the PDRS Decision Support System (DF44) within the Mission Operations Directorate and HCI designers within the Intelligent Systems Branch (ER2) of the Engineering Directorate at JSC.

Analysis of Case Study Information

The information collected during the case study was evaluated to accomplish the objectives of the case study. The first objective was to describe fault management operations using an intelligent system in aerospace operations. To accomplish this objective, a sequence of activities were performed. The resources available to the fault management team were identified, including characterizations of aerospace operations, agents and teams, and information. Types of agent activities and interaction between agents during fault management were described. Fault management activities of both the human and the intelligent computer system in aerospace environments were characterized and a typical fault management operational scenario was outlined. Based on this activity description, the information required for fault management with intelligent systems was discussed. The results of this analysis are provided in section 3.

The next two objectives were to specify preliminary recommendations and issues concerning the design of HCI for intelligent systems. These recommendations and issues were derived by analyzing the case study observations (1) to characterize the types of activities, information, and intelligent agent interaction required for fault management with intelligent systems, and (2) to identify effective methods of information management and display. Good designs from the case study were incorporated into the examples illustrating these recommendations and issues.

® - MacDraw is a trademark of Apple Computer, Inc.

® - Macintosh is a trademark of Apple Computer, Inc.

Design problems and useful design techniques were also identified. The results of this evaluation are summarized in sections 4 and 5.

The fourth objective was the evaluation of the observed design processes for ideas concerning effective delivery of guidance. Again, case study observations were evaluated to identify effective HCI design strategies. A number of issues related to integration of HCI design guidance into the design process were identified for future investigation. The results of this evaluation are presented in section 6.

Volume 2 (Malin et al., 1991) provides the detailed analysis of each application in the case study.

2.2 Intelligent Systems in the Case Study

The following applications were evaluated in the case study:

- **Real-Time Data Systems (RTDS) Applications**
Johnson Space Center
 - **Space Shuttle Guidance, Navigation & Control (GNC) Intelligent Systems**
Monitor for orbiter sensors and Reaction Control System (RCS) jets and to detect Loss of Control of vehicle during ascent
 - **Space Shuttle Instrumentation and Communications Officer (INCO) Expert System Project (IESP)**
Monitor of command and telemetry paths
 - **Space Shuttle KU Band Self Test Expert System**
Monitor of checkout procedures for KU band radar
 - **Space Shuttle DATA COMM Expert System**
Monitor telemetry recording by the onboard flight recorders and downlist of telemetry from these recorders
 - **Space Shuttle Payload Deployment and Retrieval System (PDRS) Decision Support System (DESSY)**
Monitor of the subsystems for operation of the Space Shuttle Remote Manipulator System (RMS)
- **X-29 Remotely Augmented Vehicle Expert System (RAVES)**
Ames Research Center's (ARC) Dryden Flight Research Facility
Assist real-time, ground-based trajectory control of X-29 aircraft
- **Military Aircraft Short Take Off and Landing (STOL) Real-Time Interactive Monitoring System (RTIMES)**
Edwards Air Force Base (EAFB)
Monitor of thrust nozzle control during Short Take Off and Landing (STOL) of aircraft
- **Space Shuttle Onboard Navigation (ONAV) Expert System**
Johnson Space Center
Monitor navigation state and sensors during entry

- **Space Shuttle Rendezvous Expert System (REX)**
Johnson Space Center
Monitor procedure execution during Space Shuttle rendezvous and proximity operations
- **Space Station Operations Management System (OMS) Prototypes**
Johnson Space Center
Monitor for fault diagnosis, recovery planning, and execution of procedures for OMS
- **Space Station Module/Power Management and Distribution (SSM/PMAD)**
Marshall Space Flight Center (MSFC)
Plan, schedule and manage secondary power distribution of Space Station's power resources
- **Space Station Human Interface to Thermal Expert System (HITEX)**
Joint project of Ames Research Center and Johnson Space Center
Provides user interface to the Thermal EXpert SYstem (TEXSYS) and displays information about thermal bus system
- **Spacecraft Health Automated Reasoning Prototype (SHARP)**
Jet Propulsion Lab (JPL)
Monitor health and status of multi-mission, unmanned spacecraft and ground data systems operations
- **Space Shuttle Knowledge-Based Autonomous Test Engineer (KATE)**
John F. Kennedy Space Center (KSC)
Diagnoses anomalies and controls the Environmental Control System (ECS) application
- **Space Shuttle Intelligent Launch Decision Support System (ILDSS)**
John F. Kennedy Space Center
Assists NASA Test Director (NTD) in visualizing and managing time relationships prior to launch of spacecraft

Observations from each case based on interviews and demonstrations are documented in volume 2 (Malin et al., 1991). Hardcopies of the displays from each case are also provided for many cases.

It should be noted that most of the systems investigated were under development at the time of the case study reports. As the systems continue development, the case study reports will no longer accurately represent the system. These reports are not intended as critiques or summaries of the systems, but rather as snapshots in the development of each system, which we then used to illustrate trends and issues in the design of human-intelligent system interaction.

2.3 Evaluation of the Case Study Process

The case study was an effective method for identifying preliminary guidance. Many useful ideas and examples were observed and important issues identified. However, it was the combination of the case study with the participation in an HCI design effort that was most productive. The experience gained by actually participating in a design effort greatly enhanced the utility of the case study. First-hand observation of the design process was possible. The

development of the storyboard of HCI design concepts provided an opportunity to exercise much of the preliminary guidance collected during the case study and to test ideas for the delivery of guidance. This effort contributed materially to the generation of guidance and was vital in evaluating the use of guidance in the design process.

The case study provided a good source of "raw" observations about interfacing to intelligent system. The design concepts effort provided a testbed for refining these observations into useful recommendations and meaningful issues and for testing delivery techniques. This "testbed" involvement in the PDRS DESSY design process gave first hand experience of how HCI affects design of both the intelligent system and its user interface. This experience assisted in defining difficulties with the current intelligent system design process (i.e., the HCI design problem) and in identifying what can be done to improve this process and the designs generated by this process. The "before" and "after" examples used in section 4 to illustrate recommendations for improving intelligent system design are based on the experience gained during the PDRS design effort.

Another means of testing design guidance using actual applications is the presentation and consultancy sessions planned between intelligent system developers within NASA and the conductors of the case study.

It is recommended that future use of the case study approach be combined with some "hands-on" testing of design concepts and recommendations, because these forms of guidance are preliminary and require such testing to refine them. For example, the investigation of issues identified by this study could combine a survey of relevant research ideas with evaluation of these ideas in a specific domain using operational scenarios.

Section 3

Fault Management Operations with Intelligent Systems

Fault management has traditionally been performed by teams of human operators. The introduction of intelligent systems into real-time flight support has altered this traditional mode of operation. The team now consists of human and computer agents. Activities are distributed between these agents. This distribution of activities to software agents and the resulting joint human-computer tasking represents a key change in operations. The availability of multiple agents requires allocation of tasks to specific agents and introduces the need for dynamic task allocation in situations where an agent's capabilities to perform a task are altered in real time (e.g., intelligent system fails and the human must take over its task). Also inherent in the availability of multiple agents is the need to coordinate the activities of these agents, particularly when they are working jointly on a task. These coordination activities are independent of and in addition to the fault management activities. Finally, intelligent systems represent a new source of fault management information. The use of an intelligent system increases need for innovative information management approaches to handle the already large amounts of information used for fault management. An understanding is needed of how the operator and the intelligent system must interact to manage the monitored process and coordinate shared activities.

This section provides a description of fault management operations for aerospace domains. This description is provided to orient the reader about typical fault management operations, to describe how the use of intelligent systems affects those operations, and to assist in identifying agent activities and information requirements for use in designing fault management intelligent systems. This description of fault management operations (including terminology) and the resulting specification of information and agent activities is based primarily on observations of Space Shuttle operations, discussions with Space Shuttle flight controllers (see RTDS points of contact in appendix A, especially Don Culp, Kristen Farry, and Joe Watters), and Space Shuttle flight operations support documentation (e.g., JSC, January 1989; JSC, October 1983; Farry, 1987). The emphasis on Space Shuttle is a result of the greater availability of information for an operational program such as Space Shuttle than for a program in design such as Space Station. This description is generally representative of ground operations for most space programs. The reader is forewarned, however, that it may not fully represent all aspects of flight control and is superseded by official flight support documents. Where possible, aspects unique to a specific vehicle are identified.

3.1 Fault Management Task Description

A task description is a delineation of the task goals and the actions required to achieve those goals. Fault management is the process of detecting anomalous situations and behavior, determining the causes of anomalies, and repairing anomalies for a monitored process, while maintaining safety and the ability to perform planned operations. Thus, a fault management task description will specify the goals and actions required to detect, isolate, and recover from faults in the monitored process while maintaining safety and operational capability.

A few definitions are required to understand the fault management task. Anomalies are irregular system behavior or conditions that can be caused by a variety of influences, including faults, environmental factors, operator error, etc. A fault is the cause of failure in a system, subsystem, component, or part. A failure is the inability of a system, subsystem, component, or part to perform its required function within the specified conditions, (i.e., lost capability) (JSC, January, 1989). For example, one anomaly of a Display and Control panel is that a light did not glow when expected. The resulting failure is the inability of that panel light to glow and the fault is a burned out light bulb. A mission corresponds to a set of tasks associated with

a payload or other specific objective. For Space Shuttle, a mission is usually delineated by the period on orbit. For Space Station, a mission will be delineated by the period assigned to specific mission tasks.

The first order of business in fault management is to determine if nominal, planned operations can be continued under the anomalous circumstances. If not, immediate action must be taken to minimize the impact of anomalies to safety and mission objectives. If possible, lost capability will be recovered. If recovery is not possible, mission objectives must be evaluated in light of the capability that remains.

Fault management tasks may vary between different vehicles. For Space Shuttle, the maximum contiguous time in space is less than two weeks. For Space Station, this time span jumps to 30 years. Thus the fault management task for Space Station must include considerable emphasis on recovering lost capability in space instead of just living with reduced capability until repairs can be effected post-mission on the ground.

Real-time fault management of dynamic processes differs from the more traditional forms of troubleshooting failures. Real-time fault management must be performed on-line. Often failures produce multiple, cascading disturbances (i.e., abnormal conditions) over time. Woods has characterized this type of fault management as *disturbance management*. Appendix B discusses the cognitive aspects of disturbance management and introduces a model for disturbance management that includes cascading disturbances over time.

The fault management task can be partitioned into three primary goals:

- **Monitoring and Fault Detection**
Evaluation of the status, state, and configuration of the monitored processes and associated peripheral systems (e.g., sensors, source providing power to monitored process, etc.)
- **Safing, Mission Impact Assessment, and Reconfiguration**
Determination of functionality loss due to a fault, leading to safing procedures, mission impact assessment, and reconfiguration
- **Fault Isolation, Testing, and Recovery**
Isolation (i.e., identification) of faults through testing, and recovery of lost capability

The fault management actions associated with these goals are described in detail in the following section.

Monitoring and Fault Detection

Under nominal circumstances, the state, status, and configuration of the primary monitored process are assessed continuously. This includes maintaining awareness of the status and configuration of peripheral systems that could impact the primary monitored system (e.g., sensors, systems providing required resources). Nominal, on-going operations involving the primary system are monitored to ensure that mission objectives are accomplished.

When off-nominal events occur, alarm conditions and fault indicators from the primary monitored system or peripheral systems are detected. A variety of procedures planned to respond to off-nominal events may be executed as a part of the fault management task. The execution of planned, off-nominal procedures is monitored to determine if the procedures are executed properly and to ascertain the effects of the procedure. Off-nominal procedures include safing and reconfiguration to minimize the impact of a fault on crew safety and mission

objective, testing to reduce fault ambiguity (i.e., isolate the fault), and malfunction correction to repair the fault. Monitoring of state, status, and configuration continues throughout the fault management process to detect additional changes in behavior.

Safing, Mission Impact, and Reconfiguration

Safing, mission impact assessment, and reconfiguration are actions that occur when an event with potentially negative impact occurs (i.e., an alarm condition is detected or an unscheduled procedure is requested). Safing is the process of identifying safety impacts resulting from an event and taking appropriate action to put the crew and vehicle in a safe configuration. Safing can occur automatically (e.g., fuse blows at power glitch) or can require executing safing procedures. The need for safing should be evaluated whenever an alarm condition occurs. Mission impact is the process of determining how an event affects the capability to accomplish mission objectives, both on-going and scheduled operations. Reconfiguration is the process of altering the configuration of the affected system to minimize safety and mission impacts. Reconfiguration is considered a temporary change in configuration and represents a transient state of the system under supervision.

When an alarm condition indicates anomalous behavior in the monitored process, the first response is to minimize impacts to crew and vehicle safety. If the situation poses an immediate safety threat, indicated by violation of baseline safety criteria (e.g., sudden, large drop in cabin pressure), safing procedures will automatically initiate.

Next, lost functionality and remaining capability must be identified. The anomalous behavior is translated into a functional loss in the primary monitored process or related peripheral systems. Once functional loss has been determined, the criticality of that loss must be assessed. Criticality is the importance of a capability in achieving safe operations while satisfying mission goals. Both near-term and potential, long-term safety implications must be determined. Typically, NASA defines three levels of criticality (i.e., (1) loss of life or vehicle, (2) loss of mission, (3) everything else; see section 3.4). Criticality is mitigated by the availability of redundant capability. Redundancy is the availability of equivalent, independent capability. Should crew or vehicle safety be endangered by the functional loss, procedures to minimize the risk to crew and vehicle will be initiated. Safety impact must also be assessed when an unscheduled procedure is requested.

Once the crew and vehicle safety has been ensured, the impact of the anomalous behavior on both on-going and scheduled operations is determined. Notice that the execution of safing procedures may in themselves represent a mission impact. These mission impacts are determined by evaluating the ability to complete mission objectives with the remaining capability while satisfying the mandatory flight conditions described in the flight rules (i.e., document defining nominal and off-nominal conditions and behavior and specifying the constraints of the monitored processes). Mission impacts are then minimized by reconfiguration of the monitored or peripheral systems (e.g., switching to redundant capability) or by altering the scheduled activities (e.g., delay or eliminate activities). There may be multiple options available to minimize impacts. The best option is selected by considering (1) the current configuration of the monitored and peripheral systems (including remaining redundant capability), (2) on-going operations, (3) the criticality of any suspected faults (functional or hardware) in current configuration, (4) an assessment of whether the lost capability can be re-gained, and (5) the potential for suspected faults to propagate into other systems.

Fault Isolation, Testing, and Recovery

Once crew and vehicle safety has been ensured and necessary reconfigurations have been determined, the process of identifying and correcting the fault is initiated. Identifying the fault is usually an iterative process, consisting of diagnosis to isolate a minimum set of suspected faults and testing to distinguish between faults within that set.

The first step in diagnosis is the identification of a set of suspected faults that could cause the anomalous behavior and resulting functional loss. Next, these suspected faults are compared to faults already diagnosed, to determine if a known fault could cause the current situation. Finally, since this process occurs over time and conditions may change in the interim, all faults that are not consistent with the current conditions are eliminated from the set of suspected faults.

The fault symptoms and the set of suspected faults are used to identify tests that could reduce the remaining ambiguity between faults (both hardware and functional). The identified test procedures are evaluated for their impact to safety and mission goals. If this impact is deemed acceptable, the test procedures are scheduled for execution. An important factor in determining the acceptability of these test procedures is the criticality of the functional loss (i.e., just how important is it that the lost functionality be recovered). If the impact to recover lost capability is not acceptable, mission goals must be revisited in light of the remaining capability. An altered schedule is then created to reflect these new goals.

When the time to perform scheduled test procedures arrives, the fault management team ensures that the affected system has been properly configured for testing. The results of these tests are monitored and compared to expected behavior. Execution of a test procedure alters current conditions to provide additional information about the fault. This information is then used for further diagnosis. Once testing is complete, affected systems are returned to their configuration prior to the test.

If it is possible to isolate the fault (i.e., resolve all fault ambiguity) and the fault can be repaired, the recovery process is initiated. Malfunction correction procedures are identified and evaluated for safety and mission impact. These procedures may be executed immediately or scheduled for later execution. If it is not possible to isolate the fault, mission objectives must be altered to reflect reduced capability.

When execution of recovery procedures is imminent, all systems impacted by recovery must be placed into the proper configuration. The fault management team will monitor the results of the recovery procedures to determine if they are effective in recovering lost functionality. Once recovery is complete and verified, all systems reconfigured for recovery are configured for nominal operations. If recovery procedures fail, the fault management team must evaluate remaining options for recovery.

It was observed that some faults in Space Shuttle systems with acceptable mission impact were not isolated during the mission. In these cases the fault was isolated and repaired on the ground after the mission was complete. Since the Space Station does not return to the ground, it will not have the luxury of postponing fault isolation and recovery and must deal with the issue of performing difficult maintenance tasks in space.

3.2 Team Resources and Constraints

Fault management goals and actions have been specified independent of the means of accomplishing these goals and actions. The "means" consists of the resources and constraints

of the fault management team and the flight support environment. These resources and constraints include the availability and capability of agents, the availability and reliability of information, and the attributes of the operational environment. When multiple agents are available, the interaction between agents to accomplish fault management actions must also be considered. The resources and constraints that determine specific agent activities are described below.

3.2.1 Characteristics of Aerospace Operations

Aerospace operations and the environment in which such operations occur impose constraints on the fault management task and introduce challenges for the fault management team. Literature from both aerospace operations and process control was reviewed to characterize the constraints imposed by the aerospace domain on flight support systems and personnel. Both process control and aerospace operations involve complex systems operating in high risk, real-time environments.

Real-time constraints are recognized as a key element of complex monitoring and control systems, whether in the area of process control or aerospace operations. These constraints levy performance requirements on the agents controlling these systems. Operational support must be continuous and on-line, which introduces the prospect of vigilance problems for human operators required to support for long duration periods (Buck, 1989). These problems manifest themselves in such forms as errors in observation and failure to remember pre-existing conditions (Shaw, 1988). Vigilance problems can be introduced by the intelligent system also. If the operator is not adequately aware of on-going operations (i.e., "in the loop"), it will be difficult for him to handle unanticipated situations outside the scope of the intelligent system (Buck, 1989) or take over from the intelligent system when it fails.

Changes in both the domain environment and the support environment are propagated through the monitored process in complex ways. Propagation delays can be so long that humans are unable to detect these changes or so short that human response time becomes a critical limitation. Changes in the domain environment occur even under normal operations, when physical devices alter their operating characteristics (e.g., normal operating range, stable points) with use. These evolutionary changes must be recognized and accounted for by human operators (Jakob and Suslenschi, 1990). Intelligent systems can be effective in compensating for human perceptual limitations in detecting changes occurring over very short or very long time periods.

Intelligent systems have characteristics that can affect their ability to improve flight operations support. They represent an additional information source to be managed in an environment already inundated with information. The information provided by an intelligent system is not always informative, especially when statements or recommendations are provided with no context for interpretation or clarification of the reasoning behind these conclusions. Intelligent systems are subject to errors also. They can be brittle, failing catastrophically in situations that exceed the bounds of their encoded knowledge. Thus, the intelligent system must be monitored by the human. This can increase operator workload in an already high workload environment.

The monitored systems often reside in space, which is a hostile, constrained environment for human activities (e.g., microgravity) (Rudisill, April 1990). Hostile environments complicate access to the monitored process and peripheral systems for fault management activities. Protection for humans against the hostile space environment represents a significant cost in the design of manned aerospace systems. The high cost of transporting payload to orbit also results in significant resource constraints on orbit. These constrained resources limit the number of humans that can be sustained in space, resulting in frequent ground-controlled

operations. Operation of aerospace systems is thus characterized by remoteness of control. This remoteness between the operator and the monitored process can be increased by adding an intelligent system in the monitoring and control loop, unless the HCI reinforces the operator's model of the monitored process (Buck, 1989).

Aerospace systems can also be characterized as complex systems (Jakob and Suslenschi, 1990 and Rudisill, April 1990). This includes both structural complexity (i.e., large numbers of components with multiple connections) and functional complexity (i.e., significant, diverse capability). Correspondingly, there are multiple tasks performed in parallel by multiple cooperating agents (e.g., ground flight controllers, crew onboard the spacecraft, and intelligent systems), resulting in complex operational sequences and large amounts of information required to monitor and control the monitored system (e.g., procedures, planned activities, sensed data, design specifications). Alarm systems used to detect faults in the monitored process often produce an over-abundance of related alarms that must be evaluated (synthesized) to identify the fault (Shaw, 1988; appendix B). This wealth of information and activity results in high data rates and complex mechanisms to access this information, which can overload the human operator. A human-intelligent system team can be effective in reducing activity complexity and information overload by providing coordinated multi-tasking and HCI designed to manage information.

In addition to the quantity of information required for complex, aerospace systems, the quality of the available information can complicate operations. Uncertainty exists in both the sensed data and in the models of expected behavior. Sensed measurements can be corrupted by sensor failures or degraded while being measured (i.e., limitations or mis-calibrations of the sensor can add noise or bias to the data) and while being transmitted (i.e., alteration by the transmission media). Due to the weight constraints on orbit, the number of sensors is often limited. Limited sensors can result in a lack of available information to clarify ambiguous behavior.

A variety of expected behavioral states exist for the monitored process, corresponding to operational contexts (e.g., start up, shut down, steady-state, degraded operations, normal or crisis management). Many of these behavioral states are well-understood and anticipated from models of the monitored process coupled with operational experience. Models of expected process behavior (e.g., mathematical, procedural) are inherently approximations, however, and can produce erroneous results when modeling assumptions do not match environmental characteristics. Such unanticipated situations represent uncertainty in behavioral models and must be accommodated by the fault management team.

The difficulty of making decisions in the face of uncertain information is complicated by the criticality of operations (Woods, 1986). The hostile space environment results in frequent hazardous operations. An incorrect response/decision by an operator under such conditions can have serious repercussions (i.e., threat to life or vehicle) (Shepard, 1990). The dangers inherent in such high risk situations are exacerbated by typical, frequent interruptions during critical operations (Mitchell, 1990).

There is a need to provide intelligent support systems for flight operators to assist human operators in managing the complex, highly constrained aerospace operational environment just described. The very nature of this environment, however, complicates the development of intelligent flight support systems. The complexity of the operator's task requires extensive training, often with limited training resources, resulting in a lack of available, operational expertise for development of intelligent systems (Remington, 1990). This problem is enhanced by the fact that development of large-scale aerospace projects often extend over long periods of time (Rudisill, April, 1990), posing the potential for further loss of expertise.

In summary, the challenges of aerospace operations fall into three broad categories:

- Complexity of design and operation
- High rate of dynamics
- Deficiencies in information and capability

These challenges are integral to Distributed Artificial Intelligence (DAI) Open Information Systems (OIS) as well, which are defined as "large-scale information systems [*i.e., complex systems*] that are always subject to unanticipated outcomes in their operations [*i.e., deficiencies*] and new information from their environment [*i.e., dynamics*]" (Hewitt, 1991).

Table 3-1 provides a summary of aerospace fault management characteristics that impact the development and use of intelligent fault management systems. Details of how these characteristics affect agent activities and information requirements will be discussed in later sections.

Table 3-1. Characteristics of Aerospace Fault Management

Characteristics of Aerospace Fault Management
Complexity of design and operation
• Resident in hostile, constrained environment (e.g., microgravity)
• Remoteness of control
• Complexity of engineered systems (structural and functional)
• Decisions under conditions of high risk
• Criticality of making correct response
• Continuous, long duration support periods for operators
• Multiple tasks performed in parallel by multiple operators
High rate of dynamics
• Real-time constraints and performance requirements
• High data rates due to physical dynamics
• Large amounts of information due to complexity of systems and operations
• Dynamics often outside range of human perception
• Frequent interruptions during critical operations
Deficiencies in information and capability
• Uncertainty of information (data and models of behavior)
• Unavailable information (inadequate sensors, limited bandwidth for transmission to ground)
• Limited resources onorbit, both human and expendable
• Unanticipated situations
• Decisions under conditions of uncertainty

3.2.2 Characteristics of Agents and Teams

In section 1, the concept of a joint human-intelligent system team was introduced. This team consists of an operator (onboard crew member or ground flight controller) and an intelligent system. As a team, the operator and intelligent system actively cooperate in the accomplishment of fault management tasks. This team metaphor is further qualified where the

intelligent system assists the operator. This qualification implies that the operator is in control of the fault management process. The operator makes the final decisions in fault management situations and bears the responsibility of the results of those decisions. The concept of a fault management team includes a different view of the user. Typically, the user is envisioned as an entity separate from the system. The analogy of a fault management team brings the user into the system as an integral part of the system. Both the user and the intelligent system are "agents", entities responsible to accomplish specific fault management tasks. This difference means that the user becomes an essential part of the system design, and that specification of user capabilities and responsibilities is a central design activity.

Tasks are assigned to specific agents of the fault management team based on the available capabilities and allowable behavior of each agent. The assignment of a task implies a responsibility for accomplishment of the task. Tasks should be assigned to utilize the strengths and skills of each agent and to compensate for individual weakness. When the team consists of both human agents and intelligent software agents, it is important to characterize each type of agent. For example, humans are effective at recognizing constant patterns in varying situations, so tasks requiring this skill should be allocated to humans (Wexelblat, 1989). Alternately, humans are perceptually limited (section 3.2.1). The intelligent system can easily detect events outside human perception (e.g., events occurring very fast or very slow).

Another insight into human diagnostic behavior comes from the field of decision making. Humans tend to form subjective biases to distinguish hypotheses during decision making. These biases are manifested as a tendency to (Arkes and Hammond, 1988):

- Focus on one hypothesis, which delays or eliminates consideration of alternatives
- Forget evidence to the contrary of a favored hypothesis (termed "favorable memory errors")

These biases can seriously impact the ability of an operator to make correct decisions. Yet the final decision making responsibility lies with the operator. The intelligent system does not necessarily have these biases, and can thus compensate for them by providing an unbiased evaluation of all alternatives and by reminding the operator of evidence contrary to a selected hypothesis.

Performance should be a consideration in assigning tasks. Some tasks may impose timing constraints that are outside of the boundaries of human perception (i.e., either very fast or very slow). Hardware is already used to perform some such tasks (e.g., the use of a fuse in an electronic circuit to prevent power surges from overloading a powered system). Such tasks that are not amenable to hardware solutions are prime candidates for the intelligent system. An example where software is used to meet performance requirements with minimum timing constraints is the use of software initiated safing procedures instead of manually initiated procedures. An example where software is used to meet performance requirements with maximum timing constraints is the automated detection of long-term trends in fault characteristics instead of expecting operators to notice such trends.

The reliability of the agent in performing a task and the criticality of the results of that task should also be considered when assigning tasks. The reliability of an agent can be seriously affected by the assigned workload and the number of interruptions that the agent is subject to in the course of performing a task. Tasks should be assigned such that workloads are balanced between team members and to minimize the interruption of a task by another team member. Design of the display for management of large amounts of dynamic information can reduce operator workload also.

Differences in system design and operation that are inherent in the monitored process software can affect the task assignments of the fault management team. For example, the Space Station software includes software, the Operations Management Application¹ (OMA), to coordinate the primary vehicle systems (e.g., Electrical Power System). For Space Shuttle, this coordination task is entirely a human responsibility. Thus the Space Station fault management responsibilities would include monitoring the OMA for failures as well as monitoring the primary vehicle systems for failures.

It is possible that as tasks are better defined/understood over time (maturity), nominal task allocations will shift (e.g., tend to automation over time). The system should be designed to allow such evolution.

An agent *role* corresponds to a specific set of agent capabilities and behavior. Agents may fulfill multiple roles. Agent capabilities and behavior may differ significantly in different roles (e.g., agent may be able to issue commands when fulfilling one role (real-time control) and unable in another (planning)). Agent roles are not necessarily fixed for the duration of flight support. As circumstances change, the operator and the intelligent system can alter their roles to meet the requirements of a given support situation.

A *software mode* is a manner of specifying valid activities and agent behavior by limiting the available information and software capabilities. Software modes represent a constrained approach to variable or dynamic task sharing between the operator and the intelligent system by pre-specifying the range of acceptable agent behaviors (Johns, 1990). The mode can directly indicate task allocation (e.g., MANUAL mode means the human operator executes the task) or the mode can imply the task allocation (e.g., MONITOR mode implies that the human operator does not execute the task). Modes for the intelligent system can affect the reasoning used, the available knowledge base, or how information is displayed. They represent a context for the actions of agents. A number of intelligent system modes were observed in the course of the case study. See Table 3-2 for examples of these modes.

¹ The original OMA concept has been restructured as the Integrated Station Executive (ISE).

Table 3-2. Examples of Software Modes Observed during the Case Study

Examples of Observed Software Modes

Task Modes with Implied Task Allocation

• WHAT-IF

What-if mode provides the ability for the operator to suspend or submerge into the background nominal fault management activities while evaluating the current situation under conditions of greater control than real-time modes. This mode permits the operator to conjecture a hypothesis about the current situation, alter the conditions seen by the intelligent system to match that conjecture, and observe the results of that alteration (i.e., "What would happen if?") without affecting current nominal operations. It is closely tied to simulation, prediction, and envisionment. WHAT-IF can be used to:

- Look ahead and describe the impacts of current actions and events
- Diagnose faults by postulating faults to see if they yield the current fault symptoms
- Investigate response options in unanticipated situations

• MONITOR vs CONTROL

In Monitor mode, team agents evaluate data and draw conclusions, but do not exercise control over the monitored process or the intelligent system. The results of a monitor mode are recommendations. In CONTROL mode, team agents actively participate in altering the monitored process or the intelligent system. Control may either be exercised by the operator, by the intelligent system, or both.

• REAL-TIME vs REVIEW

During REAL-TIME mode, dynamic information is used as soon as it is available (in real-time). Dynamic information can also be recorded in real-time for later use. REVIEW can consist of re-displaying information recorded previously (playback) or it can involve operating on recorded information to generate new results (replay).

Modes Explicitly Defining Task Allocation

• AUTOMATIC vs MANUAL

AUTOMATIC is a mode where the hardware or software (including the intelligent system) takes action without intervention from the operator. MANUAL mode is a mode where the operator must initiate each action taken relative to the fault management task.

• USER vs DEVELOPER vs EXPERT

Throughout the lifecycle of an intelligent system, a variety of humans with distinctly different goals interact with the system. These modes all represent types of HCI with the intelligent system that correspond to the role of the human with an associated set of goals. The USER mode represents the HCI used during real-time operation of the intelligent system. Both the DEVELOPER and EXPERT modes are off-line modes. The DEVELOPER mode is used by the software implementer to develop, test, and upgrade the intelligent system. The EXPERT mode is used by the domain expert to enter domain knowledge into the intelligent system.

3.2.3 Characteristics of Information

All information used for fault management has the following three attributes (although they are not always explicitly provided for operational use):

- **Source and Authority**
Information source is the point of origination of an information item. Authority of information is the ability of information to influence the behavior of the entity receiving the information. Information sources typically have an associated authority based on the reliability and accuracy of the information produced by the source.
- **Data Quality**
Data quality assesses the presence and severity of imperfections in the content of an information item.
- **Timing**
There are two aspects to timing of information: availability and sequence. Availability indicates access to the most current value for a parameter. Sequence refers to the order in which information is sampled for transmission.

These attributes are described in greater detail in the following discussion.

Source and Authority

The information used during flight activities comes from a wide variety of sources. The ability to distinguish where an item of information comes from is useful because it allows the agent to associate a level of confidence in the accuracy of the information or believability of the source. This confidence represents the ability of information from that source to influence agent behavior, i.e., the *authority* of the source.

The authority of a source will depend upon the role defined for the source in the fault management process and the capabilities possessed by the source. The assessment of capabilities includes a recognition of the design constraints of the data generator (e.g., a model with simplified assumptions can reduce the accuracy of generated information). Authority will be altered by the agent's experience with data from the source (e.g., this radar has a history of providing biased data due to infrequent calibration). The existence of alternate sources of information will also affect the authority of a source, since redundancy allows comparison of information items resulting in increased confidence when they agree and decreased confidence when they do not.

One way to characterize information sources is by the identity of the source. This characterization supports association of believability with source. The sources of information for fault management can be separated into four categories:

- **Sources of Dynamic Information**
(data updated in real time)
 - Vehicle and payload software and hardware
 - Intelligent system software and hardware
 - Ground support software and hardware
 - Communications software and hardware
 - Crew of vehicle
 - Ground flight controllers

- **Baseline Operations Information Sources**
 - Operations Documentation
 - Operations Expertise (e.g., Space Shuttle generic training in integrated simulations) and Experience (e.g., actual flight support)
- **Mission Specific Operations Information Sources**
 - Mission Definition Documentation
 - Mission Specific Expertise (e.g., Space Shuttle mission specific training in integrated simulations) and Experience (e.g., previous support of similar mission)
- **Sources of Design Knowledge**
 - Design Documentation
 - Design Expertise (e.g., design engineers and implementing contractors)

Another way to characterize information source is to indicate how the information was generated at the source. Method of generation often implies the types of inaccuracies that the resulting information is subject to. Four categories of information generation have been identified:

- **Sensed:** perceived by a sensory mechanism, such as a measurement device or a human

Sensed information can be degraded by imperfections in the process of perception (e.g., noise introduced by the measuring device, sample rate too slow to capture dynamics).
- **Computed:** mathematical combination of sensed information

The accuracy of computed information is dependent upon the exactness of the algorithm, the correctness of the sensed information, and the precision of the implementation environment (e.g., the hardware platform and software tool used to implement an algorithm).
- **Simulated:** information derived from a model of the source of information

The accuracy of simulated information is dependent upon the design assumptions (e.g., fidelity of model) and the precision of the implementation environment (e.g., the hardware platform and software tool used to implement simulation). Predicted information is a type of simulated information that references events expected to occur in the future.
- **Inferred:** derived by evidential reasoning

The accuracy of inferred information is affected by the reasoning strategies of the inferencer, the completeness of the knowledge base (including experience, or previous exposure to wide range of situations or behavior), imperfections in the evidence, and the simplifying assumptions.

Data Quality

Data quality is an assessment of the presence and severity of imperfections in the information content. Imperfections in information affect the accuracy of all conclusions based on the imperfect information. Imperfections in information can be introduced by the information source (as described previously in this section) or by the transmission media (e.g. noise introduced by the communication channel). Examples of data quality assessments within flight operations include statistical evaluation of data (e.g., used for selecting radar measurements)

and bit error checks (e.g., use of a code attached to each information item to detect transmission errors that alter individual bits).

The ability to use imperfect information is dependent upon the type of imperfection and the planned use of the information. In some cases it is possible to correct the imperfection or to minimize the effect of the imperfection. Post and Sage (1990) characterized imperfections by the degree of uncertainty and incompleteness that they introduced into the information. Three types of imperfect information were delineated. These types are described to clarify the considerations that affect an agent's assessment of data quality:

- **Ambiguous:** supporting multiple interpretations

Ambiguity implies a need for more information but does not apply when information normally available is missing (i.e., incomplete data set). Equivocality is intentional ambiguity to mislead or confuse the receptor of the information. Equivocality will be ignored for the purposes of civilian aerospace, although such an imperfection is quite conceivable for military applications. An example of ambiguous information is a sensor measurement that can be interpreted as either a failed component or a failed sensor.

- **Inconsistent:** conflicting or in disagreement with

Inconsistency implies that a comparison of information has occurred. This comparison can be with respect to an alternate source of the information or with respect to the same source of information evaluated over time (i.e., intermittent behavior). An example of inconsistent information occurs when measurements from two radars locate a vehicle at different positions.

- **Imprecise:** inexact or lacking accuracy

A lack of precision represents a loss of information (i.e., incomplete). An example where imprecise information occurs is when the measurement process truncates a data value, resulting in loss of precision.

The assessment of uncertainty is a topic of interest for intelligent systems. A variety of measures of uncertainty have been formulated, including probability, confidence, belief, and degree of membership (e.g., fuzzy logic). Note that the degree of uncertainty will be affected by the availability of alternate, redundant sources of information.

The assessment of completeness is dependent upon the planned use of the information. The missing information may be implicit or irrelevant, depending upon the targeted application of that information. Notice that the absence of information can be informative (e.g., the absence of data updating indicates Loss of Signal). Information that is totally missing (not just loss of precision) is considered to be unavailable. The availability of information is discussed in the following section.

Timing

Information resulting from dynamic processes have values that are updated in real time. Dynamic information is *available* when the most current value for a parameter is accessible. Information can be temporarily unavailable, or *static*, due to loss of data source. Such a loss can result from the inability to downlist due to Loss of Signal (LOS) when the vehicle is outside the range for transmission to the ground or due to failure of the transmission system. When information is static, it may be useful to indicate a range within which the static value remains valid for use (i.e., an assessment of information staleness). This validity range can be

delimited by time interval or event. Information can be permanently unavailable due to inherent limitations in transmission media (e.g., insufficient bandwidth on downlist) or simply because it is not provided for transmission (e.g., value is not stored for downlisting).

Constant information remains fixed for the duration of a mission. Constants are assumed to always be available. For Space Shuttle, there are two classes of constants, mission specific constants that are loaded prior to a mission and design constants that remain the same for all missions. For Space Station, there should be corresponding types of constants for both the payloads and the orbiter.

Sequence implies an ordering of values based on a timetag associated with each value. These timetags typically refer to the time that a value was stored for transmission (e.g., sampled or computed) and thus reflect the order in which events that influence data values occurred. The sequence of parameters can be very important when diagnosing faults in complex systems, for the order of fault symptoms is important in isolating the source of a fault.

3.3 Agent Activities

Activities are actions that have been assigned to specific agents and which employ available resources (e.g., information, capabilities) to achieve the goals supported by these actions (Lochbaum and Sidner, 1990). Typically, agent activities operate on the monitored process, or its related peripheral systems, to achieve mission goals in the presence of anomalies. These activities are called *fault management activities*. When multiple agents are working together as a team, however, it is necessary to consider a class of activities that describe how team members interact to jointly accomplish tasks. These *coordination activities* ensure effective synchronization of agents and ascertain that agents are performing as expected (Gasser, 1991).

The introduction of intelligent systems into the flight support team introduces the need for agent coordination activities between the operator and the intelligent system. An intelligent system can greatly reduce the operator workload for monitoring and control. However, it also adds another potential source of error. The operator must not only assess errors in the monitored process, he must assess erroneous behavior in the intelligent system as well. Since the intelligent system is an active participant in problem-solving, a likely dilemma is the situation where the intelligent system mis-concludes or makes an error, either through limitations in the knowledge base (i.e., brittleness, section 3.2.1; Gasser, 1991) or erroneous input. The operator should be able to prevent errors in the intelligent system by intervening to redirect its actions or alter the conditions perceived by the intelligent system. This is similar to the human interaction between an expert and a less experienced but oriented person. The ability to observe, understand, and control intelligent system behavior is important, because intelligent systems are likely to be regularly upgraded due to iterative development process, and therefore less fully tested and more likely to fail.

Two scenarios can result from an intelligent system that is not understandable or controllable: either the operator blindly follows the intelligent system or he entirely ignores it. If the operator relies too heavily on the intelligent system without evaluating its conclusions, he may become less attentive than without the intelligent system and ultimately lose necessary skills for flight support. If the operator ignores the intelligent system, then at best the operator does his job as before and flight support performance is the same as before the intelligent system was provided. At worst the introduction of the intelligent system may have removed operator access to critical information and flight support performance is worse than before the intelligent system was provided.

The concept of a fault management team consisting of human (operator) and computer (intelligent system) agents is investigated to define the activities required for multi-agent fault management. The types of agent activities necessary to accomplish fault management goals are described. Agent interaction to support joint/shared tasking is discussed. Specific activities associated with fault management are identified in section 3.5.1.

3.3.1 Types of Agent Activities

Agent activities support two objectives, managing the monitored process and coordinating agents. Both objectives require similar types of activities. Evaluation of the fault management goals defined in section 3.1 reveals that goals are accomplished by three basic types of actions, specifically (1) monitoring and assessment of situation and system/agent behavior, (2) planning of activities with dynamic re-plan at anomalies or contingencies, and (3) intervention and control to alter actions or conditions in response to situation or behavior. Table 3-3 lists these actions and associates with each action a classification of the type of activity required. These types of activities are discussed in more detail below.

Table 3-3. Types of Activities Associated with Fault Management Actions

FAULT MANAGEMENT ACTION	TYPE OF ACTIVITY
Monitoring and Fault Detection Assess state, status, operations, and configuration of the monitored process and support systems Monitor state, status, and configuration of related systems Initiate execution of procedures for nominal operations Monitor on-going operations and procedure execution Detect alarm conditions and fault indicators	Monitoring and Assessment Monitoring and Assessment Intervention and Control Monitoring and Assessment Monitoring and Assessment
Safing, Mission Impact, and Reconfiguration Identify lost functionality and remaining capability Determine potential safety implications of functional loss Initiate necessary safing procedures Assess if lost capability is retrievable Determine mission impact of functional loss and required procedures Identify reconfiguration options and schedule changes	Monitoring and Assessment Planning Intervention and Control Monitoring and Assessment Planning Planning
Fault Isolation, Testing, and Recovery Determine set of suspected faults Identify tests to reduce remaining ambiguity in suspected faults Initiate test procedures to isolate fault Monitor execution of test procedures Identify malfunction correction procedures when fault diagnosed Monitor execution of malfunction correction procedures Initiate malfunction correction procedures when fault ambiguity is resolved	Monitoring and Assessment Planning Intervention and Control Monitoring and Assessment Planning Monitoring and Assessment Intervention and Control

Monitoring and Assessment

The predominant activity of the fault management team during nominal situations is detecting significant changes in dynamic data indicating off-nominal behavior, or *monitoring and assessment* of information. Once off-nominal behavior has been detected, the assessment process continues. Functional loss must be assessed to determine what capability remains to accomplish mission goals. Off-nominal behavior and the procedures that respond to such behavior must be assessed for safety and mission impacts. Faults must be associated with symptomatic behavior.

There are two approaches to Assessment. One approach is a quick scan of important information to provide a summary view of the events associated with the monitored process, the intelligent system, and other operators. This approach is termed Walk-up Assessment. The second approach is assessment by In-depth Analysis. This approach is used when the details of some portion of the overall health and configuration or on-going operations require further investigation. Both approaches are required for flight support. Walk-up Assessment is useful when the operator must quickly orient himself to the current situation (e.g., at handover between operators). In-depth Analysis is useful when the operator must analyze the detailed data supporting the summary information or closely monitor an operational segment (e.g., isolating a fault). Both types of assessment must be performed in the presence of frequent interruptions and with deficiencies of information and capability.

A variety of methods for assessing situation and system behavior were observed during the case study. Table 3-4 summarizes some of these methods. Although these observations do not provide a complete listing of methods used for fault management, they are representative of typical approaches and are included as examples of such.

Table 3-4. Examples of Methods for Assessing Situation and System Behavior

Observed Methods for Assessing Situation and System Behavior

Detection of a problem can result from a direct comparison of the expected behavior of the monitored system to the actual behavior of the monitored system under a given set of conditions. Expected behavior may be determined from:

- Limits specified in the flight rules (values, rate of change)
- Predictions from system models (including statistical prediction)
- Expectations based on operational observation (heuristic)
- Mission specific predictions based on pre-flight studies (i.e., flight techniques)

Alternately, when direct symptoms of a problem are not available (for example, due to limited bandwidth on the downlisted telemetry stream), problems may be detected inferentially, by observing (or hearing via a communications loop) evidence of behavior that indicates the existence of a problem. An example of such indirect evidence is the observation of crew keypad entries associated with an activity to confirm that the activity occurred.

Inferential evidence may also be useful for fault isolation by elimination. Consider the example where a problem can be attributed to one of two faults, A or B. By performing a test that determines that fault A does not exist, it can be inferred that fault B exists, though no direct evidence of fault B is observed.

Redundant, independent capabilities are frequently used to provide backup capability. If redundant sources of information are available and healthy, they can be used to assist in fault detection and isolation (e.g., use of majority vote to determine which of 3 redundant IMUs have failed, use of redundant radar measurements to determine which source is in error).

Another method of fault isolation resembles the "generate, test, and debug" paradigm (Simmons and Davis, 1987) used for event reconstruction. In this method, an event (i.e., this fault is the cause of the problem) is postulated and the resulting constraints and conditions (i.e., alarm conditions (behavior) would be observed for this fault) are estimated. These expected alarm conditions are then compared to the actual alarm conditions to see if the fault could have yielded the correct failure signature. This method iterates over the space of possible causes until a satisfactory match is found or no other hypothetical causes are available. A variation of this technique, "what-if" testing, is a useful technique to explore alternatives and anticipate problems when predicting mission impact of a fault and determining reconfiguration options to minimize impact.

Planning and Dynamic Re-planning

Procedures are elements of a plan that specify the sequences of activities performed to achieve mission goals within the constraints of flight rules. Procedures relate goals to required actions. Agent activities are actions that have been assigned to a specific agent. These activities are conditioned upon access to and manipulation of specific information. Thus, procedures assist in linking task goals to agent activities and information requirements.

The planning activity for fault management is preformed in two phases. The first phase consists of a pre-mission planning activity. Prior to the initiation of a mission, mission plans are developed consisting of pre-specified sequences of procedures designed to meet mission objectives. The pre-mission planning process consists of overlaying specific mission criteria (e.g., payload constraints) onto a template of standard operational procedures. The resulting plans may simply link procedures taken directly from documents of standard, operational procedures (e.g., malfunction procedure workbook, such as JSC, April 1988) or may include additional custom procedures that meet unique mission criteria (e.g., specification of trajectory of manipulator during berthing, such as JSC, February 1989).

The second phase of planning is dynamic, real-time re-plan. Re-plan is the process of altering planned procedures in response to unexpected or contingency situations. Re-plan can be as simple as the use of operator experience to interpret a procedure (see table 3-5 below) or as complex as developing an entirely new method for performing an action.

Three approaches to the development of procedures have been observed in flight operations: (1) procedures are generated and documented pre-mission, (2) undocumented procedures are derived from previous flight support experience (i.e., operator heuristics), and (3) contingency procedures are created in real time when documented procedures and operator heuristics are not effective. Each of these approaches is discussed below.

Documented procedures are written prior to a mission for both nominal activities and off-nominal activities. For nominal situations, procedure sequences are constructed to accomplish planned activities. These activities may be standard for all missions (e.g., must always deploy the Space Shuttle manipulator arm prior to flight operations) or may be mission specific activities generated to resolve issues related to unique mission objectives (e.g., position a payload attached to the manipulator arm at a specific location for special payload operations). For off-nominal situations, procedure sequences are constructed to minimize the negative impacts resulting from behavior in the monitored process and its support systems that is outside of the limits specified in the flight rules.

A variety of off-nominal, documented procedures exist. Self-test and checkout procedures are conducted during nominal operations to detect faults. Once a failure occurs, safing and reconfiguration procedures are used to stabilize the affected system in a safe configuration. Fault isolation when fault ambiguity exists is accomplished by executing test procedures that will force the affected portions of the faulty system to exhibit distinguishing behavior. Malfunction procedures are used to isolate and recover from faults.

The second approach to procedure development is the synthesis of operational observations made by operators during multiple support situations into operator heuristics. Operator heuristics are procedures typically not written down that result from operational experience gained through training simulations and missions. Operator heuristics do not replace written procedures, but address situations not fully resolved by written procedures. Some examples of observed operator heuristics are shown in table 3-5.

Table 3-5. Examples of Typical Heuristics used in Flight Operations

Examples of Typical Heuristics used in Flight Operations
<ul style="list-style-type: none">• Delay response to alarms near start-up of a device, since they can be due to temporary misconfiguration of the device instead of a device fault.• Use design knowledge or observed trends in reliability and quality of a component or capability to interpret an alarm condition (e.g., these components rarely fail, measurements from this radar are usually noisy; for redundant capabilities, one may be higher fidelity than the other).• Use patterns of faults or alarms as an indicator of faults at a higher level of system abstraction (i.e., a fault signature, such as normal input pressure, low output pressure, and reduced outflow rate at a node indicate a leak at the node).

To clarify how an operator heuristic can modify a written procedure, consider the following example. A procedure might include the conditional:

"If alarm Q is issued,
then system Q has failed and recovery procedures should be initiated."

Application of the first heuristic in table 3-5 to this statement results in the following modified conditional:

"If alarm Q is issued and system Q was recently powered up,
then check for misconfiguration of system Q.
"If alarm Q is issued and system Q was not recently powered up,
then system Q has failed and recovery procedures should be initiated."

The third approach to development of procedures is the real-time creation of contingency procedures. Contingency procedures are required when documented procedures and operator heuristics do not resolve an anomalous situation. Since such situations are not anticipated during mission planning, they are termed *unanticipated situations*. Unanticipated situations result from behavior either inconsistent with or not specified by flight rules or operational observation and experience. Contingency procedures are developed by modifying or adapting existing procedures and operator heuristics to accommodate the new circumstances.

Intervention and Control

It is necessary to provide capabilities to agents that allow them to meet assigned task responsibilities. These capabilities include issuing control commands to the supervised system and altering the information processed by that system. The ability to redirect or issue commands to the supervised system is called *intervention and control*. For the fault management team, the operator will typically perform intervention and control.

Intervention into and control of a system under supervision (i.e., intelligent system or monitored process) is accomplished by either altering the actions of the system or forcing a condition affecting the system. These actions or conditions are altered to correct or redirect the associated processing into a more profitable path. When intervention affects the activities of multiple agents, it should be performed in a graded way, where only small changes are introduced over time. Intervention into and control of the monitored process is accomplished by executing the class of procedures that are appropriate for the type of situation (see the discussion of planning in this section).

Successful intervention and control requires the operator to thoroughly understand both the situation and the operating characteristics of the system being altered. Prior to intervention in either the intelligent system or the monitored process, the expected behavior resulting from that intervention should be closely inspected for deleterious effects. This inspection could include testing the effect of executing scheduled procedures, evaluating new or modified procedures created in response to unanticipated situations, and observing the effect of redirecting the intelligent system. Such prediction of behavior could require simulation capability. If the operator attempts an ill-advised intervention or control activity, the intelligent system could inform the operator that it is not a good idea to intervene at this point in the activity timeline. To permit recovery from unanticipated effects of intervention, intelligent system state should be stored prior to any intervention.

Intervention may not always correct the faulty behavior. It may not be possible to recover from some types of errors or the knowledge base of the intelligent system may be sufficiently incomplete with respect to the given situation to allow the intelligent system to continue its nominally allocated tasks. In such situations, the operator should be able to assume the responsibilities formerly allocated to the intelligent system. Responsibilities can be reassigned selectively or, in critical situations, a complete override (or takeover) can be performed.

3.3.2 Interaction Between Agents

For complex systems, such as those built for aerospace, the monitoring and control process results in a number of related activities that must be coordinated to accomplish a task. When multiple team members are working simultaneously on a fault management task, information from independent activities must be exchanged and discussion relating this information to system behavior must occur. Such a joint effort may be described as *collaboration* between agents of the fault management team.

The intelligent system and the operator must collaborate to accomplish joint tasks. Communication -- the exchange of information between individuals -- is a key element of collaboration. Exchange is the important concept here. Typically, intelligent systems are designed for one-way communication, where the intelligent system tells the operator status and recommendations. Intelligent systems should be constructed to allow the operator to communicate with the intelligent system as more than a data gatherer (Roth and Woods, 1989). As the team leader, control rests with the operator. The operator has the responsibility to make final decisions and to act upon those decisions. To do so effectively, he must be able to utilize the capabilities of the intelligent system and compensate for its deficiencies. He must be able to understand intelligent system conclusions and assess their correctness and applicability for a given situation. Such an exchange represents a meaningful dialogue between team members (what Fischer (1989) terms *natural communication*). Natural communication can only occur when team members have a similar model of the monitored process behavior and an understanding of the scope of each team member's expertise. Such shared knowledge represents an *implicit communication channel* (Fischer, 1989).

Models of Agent Interaction

The intelligent system and the human can collaborate (interact) in a variety of ways. It is useful to evaluate the ways that humans interact as metaphors for the ways in which the intelligent system can interact with a human. Typically, the intelligent system serves as a consultant or advisor. As an advisor, the intelligent system provides a resource for the human problem-solver. It is usually assumed that the human knows very little in the intelligent system's area of expertise and is thus a passive agent in the interaction (Woods, 1986). This resembles the

novice/expert metaphor, where information passes primarily in one direction, from the expert to the novice (Fischer, 1989). Some variations of the advisor metaphor include:

- **Critic**
The intelligent system critiques the conclusions of the operator
- **Substitution**
The intelligent systems substitutes for the human as problem-solver
- **Reminding or Broadening**
The intelligent system reminds the operator of pre-existing conditions and alternative approaches
- **Teaching**
The operator learns from the intelligent system
- **Informational**
The intelligent system serves as another information source

A more productive approach to advisory interaction is the cooperation between partial, overlapping experts (Roth and Woods, 1989). Here both the intelligent system and the operator are experts, but their areas of expertise differ. Additionally, both have a working knowledge of the expertise of the other. Thus, the human-computer role becomes that of a partial expert interacting with a specialist. As the situation varies, different kinds of expertise will be needed and the roles of partial expert and specialist will shift between the operator and the intelligent system.

Active communication between agents is a key element of the partial-expert metaphor. In his model of agent communication (Fischer, 1989), Fischer asserts that the listener must be more intelligent than the speaker. He concludes this because the listener bears the responsibility of achieving an understanding of the implications of the situation while the speaker already understands those implications. Such considerations emphasize the importance of HCI for effective intelligent systems.

An extensive discussion of styles of collaborative interaction between human and intelligent software agents is provided in Appendix C.

Coordination Activities

The coordination activities introduced in section 3 are the means of achieving effective collaboration between agents of the fault management team. They include such activities as re-allocation of responsibility between agents, re-direction of an agent in error, or override of a malfunctioning intelligent system. Only by examining the interaction between agents and the expected activity sequences is it possible to identify coordination activities.

Tasks can be accomplished either by a single agent or by multiple, cooperating agents. When multiple agents are assigned to perform a task, they must coordinate their individual activities and collaborate to exchange necessary information. An important aspect of agent coordination for shared tasking is the allocation of responsibility for portions of a task to specific agents. The activities supporting these sub-tasks may be independent or they may have inherent dependencies (e.g., required exchange of information). A full specification of agent interaction during shared tasking requires defining these dependencies between joint agent activities and the handovers of responsibility that must occur during joint performance of activities.

Coordination of agent activities will involve some combination of the following modes of task-sharing:

- **Supervisory**
Some agents perform activities while other agents oversee the performance of these activities to guarantee that task goals are satisfied as well as possible.
- **Simultaneous**
Multiple agents perform activities in parallel, with possible dependencies between activities.
- **Sequential with Handovers**
Multiple agents perform activities serially, with possible dependencies at handover of responsibility between agents.
- **Stand-alone**
A single agent performs all activities required for a task with no dependencies on the activities of other agents.

This model is based in part upon work done for teleoperations (Backes et al., 1990).

Underlying all activities that require interaction between agents is the need for low-level support capabilities to coordinate cooperating agents (MDSSC, 1989). These support capabilities include such activities as synchronizing agents and archiving and reviewing information.

Explanation is another important collaborative capability. Recommendations concerning explanation with intelligent systems are provided in section 4.1.2. Coordination activities are described in greater detail in section 3.5.1.

3.4 Fault Management Scenarios

In section 3.3, the agent activities associated with fault management were specified. These activities are illustrated in a simple operational scenario for fault management shown in Table 3-6. An operational scenario is a description of a realistic sequence of events during a segment of flight operations. For this study, the fault management segment will initiate with the detection of anomalous behavior and conclude with the restoration to nominal operations. This simple scenario will be considered the Baseline Fault Scenario. It will be used as a starting point to characterize the more complex fault scenarios that are encountered during actual flight operations. Note that nominal monitoring of systems and operations unrelated to the anomalous behavior will continue in parallel to the fault management activities outlined in the Baseline Fault Scenario.

Typically, a fault scenario would include the assignment of activities to specific agents and the specification of how agents will interact to perform those activities. In the Baseline Fault Scenario, however, these assignments have not been made to avoid overlaying a specific architecture (and implicit application) on a general description of fault management activities. The intent of providing the Baseline Fault Scenario is to orient the reader about the nature, scope, and sequencing of fault management activities. During design, a detailed scenario specific to the application would be developed that includes activity assignments and agent interaction specification. Such a scenario would be used to test the HCI design both for the ability to effectively perform fault management activities and the ability to coordinate multiple agents who are simultaneously performing these activities.

Table 3-6. Baseline Fault Scenario

Baseline Fault Scenario

Detection of Failure

- Monitor state, status, and configuration of primary monitored system and peripheral systems
 - Monitor status of on-going operations with respect to planned mission activities
 - Observe anomalous behavior in monitored system
- Note: monitoring incoming data for symptoms of anomalous behavior will continue throughout the fault management process

Safing

- If behavior represents immediate safety threat, then automatic safing
- Determine potential, long-term impacts of behavior to crew and vehicle safety
- Perform safing procedures necessary for long-term safety

Mission Impact Assessment and Accommodation

- Determine functional loss due to anomalous behavior
- Determine impacts of functional loss to accomplishment of mission goals
- Minimize mission impacts by
 - Reconfiguration monitored process or peripheral systems
 - Delay or elimination of scheduled activities requiring lost functionality

Diagnosis

- Isolate anomalous behavior to specific sub-systems within the monitored process
- Determine list of suspected faults that would cause anomalous behavior
- Compare suspected faults to known faults
- Continue to diagnose and test until fault is isolated to a replaceable or repairable unit

Testing

- Identify tests to distinguish between faults in suspect list
- Evaluate safety and mission impacts of proposed test procedures
- Select test procedures with acceptable safety and mission impact
 - Criticality of the lost functionality will affect the definition of "acceptable impact"
- If no test procedure has acceptable impact, alter mission goals to reflect remaining capability
- Schedule test procedures
- When time to execute test procedures, configure affected systems for testing
- Initiate test procedures
- Monitor behavior observed during tests with respect to expected behavior
- Eliminate some suspected faults based on the results of tests
- Reconfigure for nominal operations after testing

Recovery

- Identify repair or recovery options
- Evaluate safety and mission impacts of proposed recovery procedures
- Select recovery option to minimize impact to safety and mission
 - Criticality of the lost functionality will affect the definition of "acceptable impact"
- If no recovery procedure has acceptable impact, alter mission goals to reflect remaining capability
- Schedule recovery procedures
- When time to execute recovery procedures, configure affected systems for recovery
- Initiate recovery procedures
- Monitor recovery procedures to verify that functional capability is restored
- Reconfigure for nominal operations after recovery

In the description of the Baseline Fault Scenario, the characteristics of the anomalous behavior were not specified. There are multiple types of anomalies that would initiate fault management activities. Anomalies identified during the course of the case study include:

- Failure of monitored system, sub-system, or component
 - During test and checkout
 - During use in mission operations
- Failure of sensor
 - During test and checkout
 - During use in mission operations
- Failure of peripheral system, sub-system, or component that degrades the primary system, sub-system, or component
- Error in timing or synchronization of systems
- Poor data quality (e.g., noise, bias)
- Onboard crew error (e.g., enter wrong key strokes)
- Ground operator error
- Intelligent system error or failure
- Misconfiguration of the system under supervision

The Baseline Fault Scenario represents a simplified case of the types of situations that actually occur during flight operations. Possible complications to this basic scenario include:

- Interruptions unrelated to fault management operations
Actual fault management operations rarely follow the sequential steps illustrated in the Baseline Fault Scenario. In the complex environment of aerospace operations, many activities are on-going simultaneously. The fault management team must continue to assess behavior of all sub-systems within the monitored process, execute operations unrelated to the current fault situation, coordinate with other members of the fault management team as well as with the control teams of the other primary vehicle systems, and handle sporadic events of the support environment (e.g., phone call, CRT fails). These activities represent interruptions to the fault management tasks that must be accommodated with minimum impact to the accomplishment of those tasks.
- Loss of data
The types of problems that can result in unavailability of data are (1) failures in the data transmission systems (e.g., loss of downlist to ground, Data Management System failure onboard Space Station, Display and Control failure on Space Shuttle), (2) Loss of Signal (LOS) outside regions of acquisition, and (3) change in transmission mode resulting in reduced bandwidth of downlist that limits the parameters transmitted to a subset of normal downlist (for ground only). Most of these problems affect only the ground support systems. For Space Station, however, the use of distributed processing onboard the vehicle introduces the possibility of loss of data due to failure in the data distribution system.
- Intermittency of anomalous behavior
In the Baseline Fault Scenario, the anomalous behavior remains consistent throughout the fault management process. It is possible for the anomalous behavior to be sporadic and only appear intermittently, however (e.g., loose wire that makes contact to short a circuit erratically). Intermittency of anomalous behavior significantly complicates the fault management task, since the unique characteristics of circumstances resulting in the anomalous behavior must first be isolated.

- **Combinations of anomalous behavior**
Anomalous behavior is not guaranteed to be caused by a single type of anomaly. Combinations of anomalies can complicate the fault management task by changing the observed symptoms of the anomalies.
- **Availability of redundant capability**
A common risk management technique in aerospace operations is the provision of redundant capability for use when primary capability is lost. Weight and size constraints onboard the vehicle limit the use of redundancy, however. Additionally, multiple faults can result in loss of redundancy. The lack of redundant capability will affect the criticality of a functional loss and affect the importance of fault recovery. These considerations will alter the priorities of the fault management process.
- **Criticality of functional capability or hardware item**
Hardware items and functional capabilities associated with flight operations are each assigned a criticality rating. This rating usually reflects the importance of the capability or hardware to safety and mission goals. For Space Shuttle, the Critical Items List (CIL) are rated as follows (JSC, January 1989):
 - Criticality 1: loss of life or vehicle
 - Criticality 2: loss of mission (specifically, first failure results in loss of mission, next related failure results in loss of life or vehicle)
 - Criticality 3: all other potential effects of failure
 Since the loss of a redundant item can affect the criticality of a item, the following special rating exist for redundant items (JSC, January 1989):
 - Criticality 1R: redundant hardware, all of which, if failed, could cause loss of life or vehicle
 - Criticality 2R: redundant hardware element, all of which , if failed, could cause loss of mission
 An important factor in determining the acceptability of test and recovery procedures is the criticality of the functional loss associated with the anomalous behavior.
- **Unanticipated anomalous behavior**
Although procedures defined prior to a mission address the majority of the situations that arise during operations, there is the possibility of anomalous behavior that was not anticipated from the design of the monitored process or observed during previous operations. The occurrence of unanticipated anomalous behavior complicates the fault management task, because procedures have to be altered during the course of the mission (see section 3.3.1 concerning re-plan). This alteration can take the form of overriding a scheduled procedure or executing an unscheduled procedure. Tables 3-7 and 3-8, respectively, provide examples of these special situations.

Table 3-7. Examples where Scheduled Procedures are not Performed

<p>Examples of situations where a scheduled procedure might not be performed:</p> <ul style="list-style-type: none">• In some circumstances, the scheduled procedure may be overridden to accommodate special cases based on observation and flight experience (e.g., ignore an alarm at start-up, since it results from the device initializing in the wrong configuration; later procedures will reconfigure the device).• Procedures are written for the worst-case scenario and may be unnecessarily drastic in situations better than worst-case.• It may not be certain whether a procedure applies to the existing situation and the fault management team must judge whether to execute it or not.• In light of the remaining capability due to multiple faults, the specified procedure results in unacceptable mission or safety impacts.

Table 3-8. Examples where Unscheduled Procedures are Performed

<p>Examples of situations where an unscheduled procedure might be performed:</p> <ul style="list-style-type: none">• The scheduled procedures did not correct problem but the criticality of the functional loss requires that some action be taken.• The fault management team anticipates the need to solve a problem due to the potential for a future fault to result in significant capability loss• A scheduled procedure is altered or a new procedure devised to meet a unique or unanticipated situation.

3.5 Fault Management Activities and Information

The types of agent activities required for multi-agent fault management and the types of information used during fault management are identified in this section. This description is based on observations from the case study, especially discussions and informal documents provided by the PDRS flight control group (i.e., K.Farry, J.Watters, and D.Culp listed in Appendix A). Additionally, the authors relied on flight control documents, including JSC (October 1983), Farry (1987), JSC (February 1988), JSC (April 1988), JSC (December 1988), JSC (January 1989), JSC (February 1989), JSC (December 1989), and JSC (June 1990) and one author's experience as a Space Shuttle flight controller (i.e., D.Schreckenghost). See appendix D for a detailed description of agent activities and information for fault management.

3.5.1 Agent Activities for Fault Management

Three types of agent activities were outlined in section 3.3.1:

- Monitoring and assessment of situation
- Planning and dynamic re-plan
- Intervention and control to alter actions or conditions

Specific fault management activities associated with these types of agent activities are described in this section. Additionally, Coordination activities required for task sharing between multiple agents are described. Appendix D provides definitions and discussion for each type of activity.

These activities represent the range of activities required for agents of the fault management team to achieve fault management goals. The activities required for a specific fault management team would be some subset of this range, dependent upon the scope of the task, the selected methods, the available agents, and the resources of the actual flight support environment.

Monitoring and Assessment

Monitoring and assessment are performed to detect significant changes in dynamic information indicating off-nominal behavior or situations. Table 3-9 summarizes the monitoring and assessment activities required for space flight operations.

Table 3-9. Agent Activities Supporting Monitoring and Assessment

Monitoring and Assessment
<ul style="list-style-type: none">• Assess state, status, health, and configuration to determine behavior of monitored process, related peripheral systems, and the intelligent system• Compare on-going operations to planned activities• Monitor the activities of the fault management team (intelligent system, crew, ground controllers) for errors and anomalous behavior• Assess the accuracy and applicability of intelligent system conclusions• Distinguish between nominal behavior and anomalous behavior• Relate anomalous behavior and failures to loss of functional capability and hardware items• Relate loss of functional capability to resulting impact to safety or mission goals• Assess ability to perform scheduled mission activities based on impacts and remaining capability• Identify possible faults resulting in observed anomalous behavior• Assess the reliability and quality of information from both the monitored process and the intelligent system• Assess the availability of data

Planning and Dynamic Re-Plan

Planning is the process of specifying desired mission activities prior to the mission. Dynamic re-plan is the real-time process of altering pre-mission plans in response to unexpected or contingency situations. Table 3-10 summarizes the planning and dynamic re-plan activities required for space flight operations.

Table 3-10. Agent Activities Supporting Planning and Dynamic Re-Plan

Planning and Dynamic Re-Plan
<ul style="list-style-type: none">• Assess potential for a failure to propagate and affect other portions of the monitored process or associated peripheral systems• Determine reconfiguration options or schedule changes to minimize impact of anomalous behavior• Determine options to reconfigure the intelligent system when in error• Assess the impact of operator intervention into or take over of either the monitored process or intelligent system prior to taking action• Predict the anomalous behavior associated with a fault• Explore alternatives for reconfiguration and schedule changes after functional loss• Explore alternatives for recovery after identification of fault• Assess recovery options based on identified fault

Intervention and Control

Intervention and control is the ability to redirect or issue commands to a supervised system. Table 3-11 summarizes the intervention and control activities required for space flight operations.

Table 3-11. Agent Activities Supporting Intervention and Control

Intervention and Control
<ul style="list-style-type: none">• Control the Intelligent system• Redirect the intelligent system• Control the acquisition of data• Command monitored process• Correct spurious or erroneous information from monitored process• Initiate scheduled procedures• Alter scheduled activities in response to fault management activities• Establish alternate modes of operation for both intelligent system and monitored process• Select between alternative solutions for<ul style="list-style-type: none">- Reconfiguration and schedule changes after functional loss- Recovery after identification of fault• Takeover of intelligent system responsibility by operator• Takeover of monitored process (crew command, uplink)

Coordination

Coordination activities are activities required for agent cooperation and communication during joint tasking. Table 3-12 summarizes the coordination activities required for space flight operations.

Table 3-12. Agent Activities Supporting Coordination

Coordination
<ul style="list-style-type: none">• Clarify intelligent system actions and conclusions by inspection of relevant information at intermediate steps of the reasoning process• Review of algorithms, processes, and reasoning strategies used to compute or derive information about the intelligent system and the monitored process• Evaluate data trends and system performance for both intelligent system and monitored process• Clarify agent responsibilities and task assignments• Review previous actions of fault management team in the context of current events• Synchronization of dependent activities (e.g., wait for a result)• Archival of important information and provision for access to that information at a later time• Manipulation of archived information from both the monitored process and intelligent system• Alteration of the appearance of the display to meet information needs

3.5.2 Fault Management Information

In the discussion about sources of information in section 3.2.3, four categories of information were identified:

- Dynamic Information
- Mission Specific Information
- Baseline Operations Information
- Design Knowledge

Specific information associated with each of these categories is delineated below. Appendix D provides definitions and a discussion of each of these types of information.

Dynamic Information

Dynamic information has values that are updated in real time due to changes in the process generating the information. There are a wide variety of information types that can be classified as dynamic. Table 3-13 provides a summary of the major types of dynamic information.

Table 3-13. Types of Dynamic Information

Dynamic Information	
• Sensed and Processed Data	
• Time	
• System State, Status, and Configuration of monitored process, peripheral systems, and intelligent system	
• Alarms, Faults and Failures, including suspected faults and fault ambiguity groups	
• Mission Impacts and Failure Propagation Potential	
• Context of On-going Operations	
• Go/No Go Calls	
• Functional Capability	
• Operator Inputs to monitored process, data acquisition system, or intelligent system	
• Performance of the monitored process, related peripheral systems and the intelligent system	
• Failure trends and performance trends in archived data	
• Dynamic intelligent system information	
- State, status, health, and configuration	
- Internal states	
- Hypotheses and intermediate or alternative solutions	
- Activities (previous, current, planned)	
- Reliability assessment	
• Archived Information	

Mission Specific Information

Mission specific information includes all information that is unique to the support of a particular mission. Table 3-14 summarizes the types of mission specific information used in space flight operations.

Table 3-14. Types of Mission Specific Information

Mission Specific Information
<ul style="list-style-type: none">• Mission Goals• Schedules and Activity Timelines• Mission Constraints• Mission Specific Procedures• Mission Configuration• Mission Specific Expertise and Experience

Baseline Operations Information

Baseline operations information includes all operations information that is required for generic flight support, not including information required to meet specific mission requirements. Baseline operations information does not change as mission goals change. Table 3-15 summarizes the baseline operations information used in space flight operations.

Table 3-15. Types of Baseline Operations Information

Baseline Operations Information
<ul style="list-style-type: none">• Flight Rules• Procedures<ul style="list-style-type: none">- Nominal activities- Test and Checkout- Safing- Reconfiguration- Malfunction Diagnosis and Correction• Modes of Operation• Operational Performance Requirements• Failure Signatures• Operations Expertise and Experience

Design Knowledge

Design knowledge is the collection of information that specifies the design of the monitored system, its peripheral systems, and the intelligent system. Table 3-16 summarizes the types of design knowledge used for space flight support.

Table 3-16. Types of Design Knowledge

Design Knowledge
<ul style="list-style-type: none">• Monitored System Specifications• Intelligent System Specifications• Design Constraints• Design Performance Specifications• Failure Modes and Effects• Redundancy• Criticality Ratings• Design Expertise

Section 4

Human-Computer Interaction Design for Intelligent Systems

This section provides design guidance in the form of recommendations and issues related to HCI design for intelligent systems. This guidance supports:

- Coordination and management of intelligent systems
- Fault management
- Information management and display

A section is devoted to each of these topics. Within each section, recommendations and issues are grouped under specific topics.

Background for the Examples

Many of the recommendations contained in this section are illustrated by examples. Examples are a powerful tool of communication. One of the dangers, however, of using examples to illustrate a general concept is that the example will be interpreted too literally while the point is missed. These examples are provided as illustrations of information problems encountered during fault management in complex domains. We are NOT recommending that the designer adopt specific designs or the "look and feel" of the user interface shown in these examples, but rather hope the designer will apply the more general concept to his own application. Rare is the development project that must not perform trade-offs and make design compromises based on cost and performance constraints. Many of the designs used in the examples represent a large leap away from the way most intelligent systems are designed today. If the designer cannot make that leap, we hope he will consider a small step in that direction.

Each example consists of two figures. The first figure illustrates the problem BEFORE the recommendation is applied. The second figure illustrates the improved human-computer interaction AFTER the recommendation is applied. The reader can compare the two figures to observe the effect of using the recommendation. This comparative format emphasizes what has been altered by use of the recommendation and how this alteration improves human-computer interaction.

All examples are based on a fault management intelligent system for a space-based Remote Manipulator System (RMS). This intelligent system is hypothetical, but is based on our work with the PDRS Decision Support System. A single domain has been selected for use throughout this section to minimize the reader's effort in comprehending the domain problems used to illustrate the recommendations. Domain information required to understand the examples is provided below.

Four scenarios have been defined for use in the examples. All are derived from post-mission reports of anomalies observed by the PDRS flight controllers (JSC, 1988). Actual events have been used as the basis of these scenarios because they accurately reflect the problem complexity and information dynamics and deficiencies of this distributed, multi-tasking environment. The types of problems experienced by fault management operators and the tasks that fault management intelligent systems are required to perform can be realistically represented in such scenarios. The four scenarios selected for use in the examples are described below:

- **Scenario 1: False Alarms in RMS Powerup Prior to RMS Operations (STS-3 and STS-4)**

This scenario is used to illustrate alarm management.

The RMS Select Switch has 3 possible settings: (1) OFF - power off to all RMS, (2) PORT - power to RMS mounted on left side of vehicle, (3) STRBD (shortened form of starboard) - power to RMS mounted on right side of vehicle. Since the RMS is typically mounted on the left side of the vehicle, the RMS is powered up by selecting the PORT setting on the RMS Select Switch.

In this scenario, the RMS is mounted on the left side of the vehicle. When the RMS is powered up, an alarm is immediately issued from two wrist joints (i.e., pitch and yaw). This alarm results from a delay (exceeding nominal 1 second) in transmitting the powerup signal. This delay is caused by the low temperature and is considered a false alarm.

- **Scenario 2: Undocumented Malfunction Procedure in Deployment of the RMS (STS 41-G)**

This scenario is used to illustrate situations where the intelligent system must be redirected or informed of information and activities not in its knowledge base.

The deployment of the RMS occurs in two stages. First the arm is rotated away from the Space Shuttle body using the Manipulator Positioning Mechanisms (MPMs). This puts the arm in the DEPLOY state. Next, the attachments (Manipulator Retention Latches or MRLs) that secure the arm to the Space Shuttle are released. The arm is then considered fully deployed (MPMs deployed and MRLs released).

The MPMs have dual, redundant motors located at the point where the arm is permanently attached to the Space Shuttle (shoulder). A deploy with two motors takes 34 seconds and a deploy with a single motor takes 68 seconds. There are three MRLs (fore, mid, and aft). Each MRL has two motors. A release with two motors takes 8 seconds and a single motor release takes 18 seconds.

In this scenario, the deploy indication is lost. Operators have observed this failure on previous missions and attribute it to maladjustment of the sensor that detects deployment. A re-command of the deploy will reset this sensor and the deploy indication will be reacquired. This is an undocumented procedure, however, which an intelligent system would not have in its knowledge base. Instead, the intelligent system recommends the initiation of malfunction procedures, based on its list of suspected faults.

- **Scenario 3: Situation-specific Violation of Malfunction Procedures during RMS Operations (STS 51-I)**

This scenario is used to illustrate impact assessment and violation of procedures based on unique conditions.

There are three joints on the RMS: shoulder (point of attachment with the Space Shuttle), elbow, and the wrist (point of attachment with the End Effector, the device connected to the payload). Four control modes are available to move these joints: (1) automatic (AUTO), (2) single-drive joint control (SINGLE), (3) direct-drive joint control (DIRECT), and (4) backup (BACKUP). Nominally, AUTO mode is used. BACKUP mode is the most constrained and is thus usually considered the least desirable. Other modes are selected based on the planned operation.

In this scenario, a power failure has resulted in the loss of all control modes for the elbow joint except BACKUP. The use of BACKUP mode for elbow joint control represents a potential impact to operations, however. If the power failure is due to a shorted motor winding, use of the BACKUP mode for the elbow will fail the entire BACKUP system.

Based on specified malfunction procedures, other joints (i.e., shoulder and wrist) would be operated in SINGLE mode. In this case, however, these malfunction procedures are violated because of additional constraints in this situation (i.e., position hold constraints resulting from the planned operation to open a sunshield on a deployed satellite). The operator selects DIRECT mode for shoulder and wrist, since it results in improved elbow joint position hold.

- **Scenario 4: Bad Sensor Data during Temperature Monitoring of Stowed RMS (STS 41-D)**

This scenario is used to illustrate alarm management.

While the RMS is stowed, the temperature of joints are monitored to ensure that temperatures do not exceed normal limits. Three sets of joints are monitored (1) pitch and yaw at the shoulder, (2) pitch at the elbow, and (3) pitch, yaw, and roll at the wrist.

In this scenario, redundant sensors are available for each joint (called sensors 1 and 2). An error in assembling the wrist roll sensor 1 has resulted in it constantly reading 71 degrees. The operator detects the bad measurement from wrist roll sensor 1, since measurements from all other sensors indicate the more normal temperature range of 30 - 40 degrees. The bad sensor will result in a false alarm, however, which the intelligent system must interpret.

Figure 4-1 illustrates the format used for BEFORE and AFTER examples. Two figures are provided, the first illustrating the BEFORE case and the second illustrating the AFTER case. Within each figure, sequences of events are often illustrated, with time increasing from left to right (e.g., Step 1 at time 1, step 2 at time 1+n, etc.). Above each time point (indicated by "STEP" at the bottom), the important events, recommendations, and display changes occurring at that time are shown. The following information is provided for each example (refer to figure 4-1):

- Summary of events and actions by operator and intelligent system (e.g. STEP 1 The intelligent system)
- Timeline illustrating events detected by the intelligent system ("Intelligent System Conclusions") and recommendations made by the intelligent system ("Intelligent System Recommendations")
- Windows from the user interface illustrating important information (e.g., data, operator actions, etc.) and demonstrating how the display changes over time

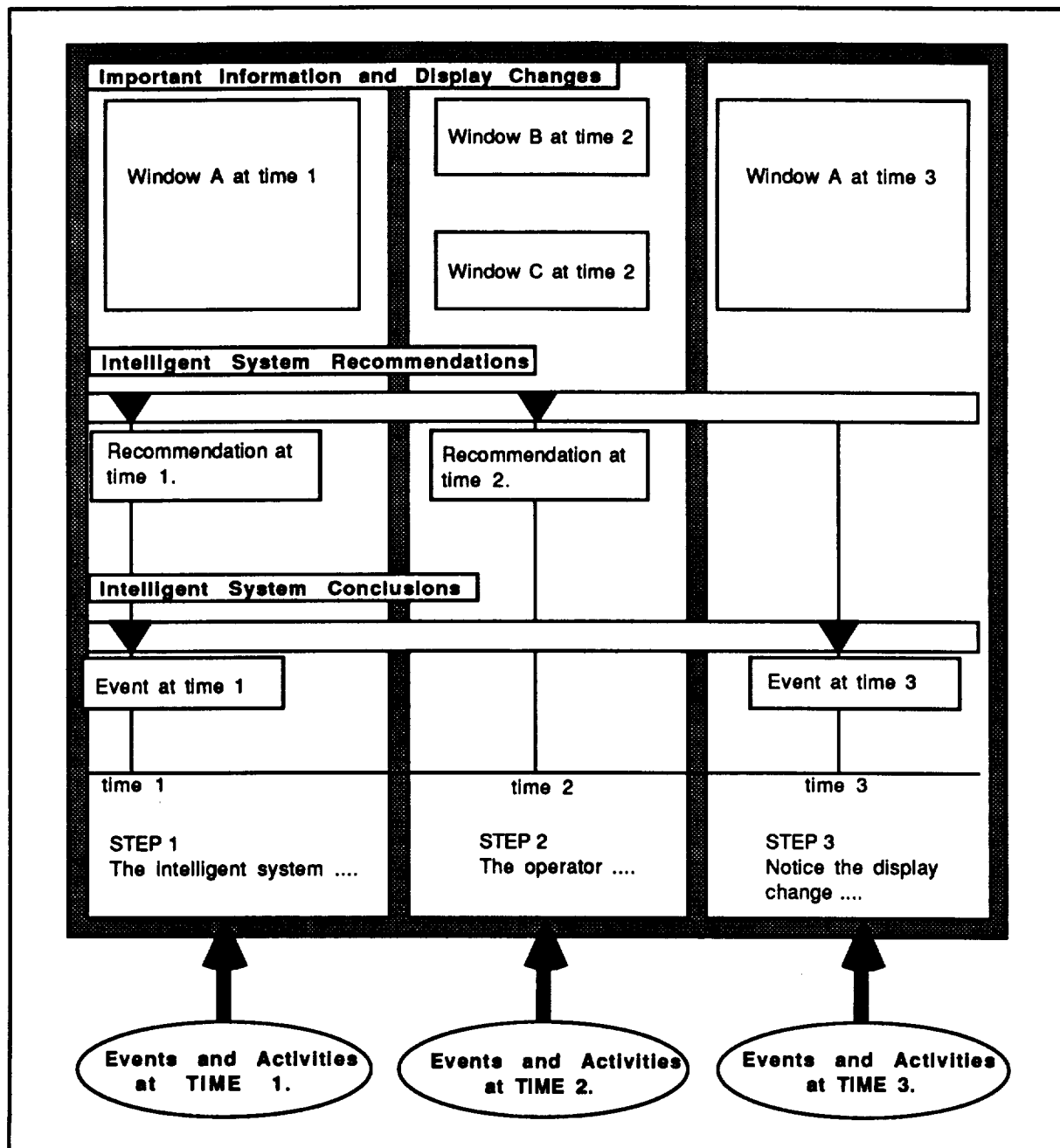





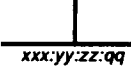

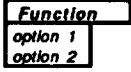





Figure 4-1. Illustration of Format Used in the BEFORE and AFTER Examples

Examples have been designed with an effort to use a consistent style in the presentation of information. The user interface conventions used in this section are summarized in table 4-1.

Table 4-1. Symbols Used in the Examples

SYMBOL	DESCRIPTION	SYMBOL	DESCRIPTION
	Mouse-selectable Control Button		Scrollable Region
	Status Region, not mouse-selectable (can be color-coded)		Event Message on Timeline
	Radio button (if selected black, else white)		Time Scale, Greenwich Mean Time
	Warning Message		Mouse-selectable Menu Functions
	Region that has been mouse-selected		Paper document
	Break in timeline, indicating extended time interval		

This concludes the discussion that provides background for the examples. Examples are used throughout Section 4. Refer to this section when questions arise about the scenarios used in examples or the format of examples. The remainder of Section 4 discusses specific design recommendations and issues.

4.1 Support for Coordination and Management of Intelligent Systems

An important conclusion of this study is the need to design intelligent systems to perform activities in coordination with humans. To assist in identifying recommendations for building such systems, the concept of a fault management team was investigated. Both the human operator and intelligent system function as agents within this team. Tasks are distributed among agents and can be reallocated if necessary. Tasks may be jointly performed by both the human and the intelligent system. As the team leader, however, the human has the final decision-making authority and retains control of the intelligent system.

Design recommendations for coordinating multiple agents as a fault management team are discussed in this section. Design issues requiring further study are also identified. Distributed tasking between multiple agents is addressed, including dynamic assignment of tasks for handling contingencies and balancing agent work load. Collaboration between agents of the fault management team is defined as the process of information exchange between agents to establish a consistent viewpoint of a situation or event. The management of erroneous intelligent system behavior is investigated, including both detection of erroneous behavior and methods of compensating for the negative effects of this behavior.

4.1.1 Multi-tasking and Dynamic Task Assignment

Integral to the the fault management team concept is the distribution of tasks between human and computer agents. Typically, the fault management team will consist of multiple human operators as well as intelligent software agents. In such an environment, tasks may be performed jointly or independently. As the situation mandates, tasks can be reallocated to different agents.

The objective of this section is to assist in designing systems for multi-tasking by multiple, intelligent agents. Recommendations address both the initial allocation of tasks to agents as well as changes to these allocations that can occur during the support period. The allocation of tasks to specific agents is related to the capabilities of each agent. Dynamic re-assignment of tasks is discussed, including the use of software modes to implement variable task assignments. Task re-assignment as a method of responding to contingency situations is investigated. Handover of task responsibility between agents during both nominal operations and contingency operations is considered. Changes in either the monitored process or the intelligent system may alter agent task priorities. Interruption of on-going activities caused by new information or activity requests is discussed.

Task Allocation

Human problem-solving can vary significantly from computer problem-solving, due to the different capabilities of each. To avoid inappropriate task allocation, the strengths and weaknesses of both humans and computers should be considered when allocating responsibility for fault management tasks (Wexelblat, 1989). In section 3.2.2, the characteristics of human and intelligent system agents were discussed in relation to agent task assignment.

Tasks allocated to the intelligent system should either amplify the capability of the operator or compensate for an inability. Task allocation should attempt to keep the operator in the loop for all decisions and important tasks. Should the intelligent system become unable to perform the tasks assigned to it (e.g., due to failure in the intelligent system), the operator must be able to take over its tasks. Thus, the operator must remain aware of on-going events and activities and be cognizant of the entire fault management process.

Tasks should also be allocated to support on-the-job training of operators. In-depth understanding of the monitored process is often achieved as a by-product of viewing information required to perform monitoring and fault management tasks. Task allocation should support development of the following types of operator expertise (Bloom, 1991):

- Understanding fault management goals, tasks, and procedures
- Developing models of process dynamics and process control systems
- Acquiring skills to interpret state information from data and decide a course of action based on that state information

The task allocation should balance the need to provide for on-the-job training with the need to amplify or compensate for operator capabilities.

Problem: Task Allocation with Task Sharing

Recommendation: Tasks should be allocated to utilize the strengths and skills of each agent of the team (human and intelligent system) and to compensate for their weaknesses. Redundant allocation should be provided to allow for shifting a task allocation to another agent (e.g., intelligent system has primary responsibility for a task with the operator as backup). On-the-job acquisition of expertise by the operator should also be considered when allocating tasks.

Example: An example of inappropriate allocation to the operator is illustrated in figure 4-2. The activities for the monitoring illustrated in figure 4-2 are reallocated in the example of figure 4-3 to include assistance by the intelligent system. Both examples are based on Scenario 2 described at the beginning of section 4.

Human characteristics affecting task allocation are related to perceptual and cognitive traits. Computer characteristics affecting task allocation are related to processing traits, including processing speed, robustness of knowledge base, and reliability of reasoning strategy. Identification of the appropriate task assignments for coordinated human-intelligent system activity requires further study.

Issue: Additional research is needed to determine appropriate task assignments for the intelligent system and effective partitions of responsibility in task sharing between the operator and the intelligent system (Johns, 1990; Shalin, 1990).

Most intelligent systems in the case study performed monitoring tasks for detection and diagnosis of failures. Procedure monitoring was observed in some cases as well. Less common allocations were control of the monitored process and agent activity planning. See Volume 2 (Malin et al., 1991) for a description of the tasks performed by the intelligent systems from the case study.

Regardless of the role, however, the human should be consulted when the potential impacts of a decision threaten mission or crew safety (with the exception of built-in hardware safety features) or when these impacts are uncertain. Intelligent systems often fail completely in situations outside the scope of their encoded knowledge. Humans perform much more robustly at the boundaries of their knowledge and can rely on a significant substrate of related knowledge and experience in making decisions.

Problem: Task Allocation with Task Sharing

BEFORE - Example Illustrating Problem:

In this situation, the MPM of the Space Shuttle RMS is being deployed. The operator is responsible for monitoring the deployment for problems and for ascertaining when the deployment is complete. The figure shows the activities that the operator must perform for this monitoring task. Notice how the operator must closely watch the display to detect state changes and elapsed time. This degree of vigilance represents an unnecessary load on the operator. A lapse of attention (such as an interruption) at the wrong time could result in the operator missing important events.

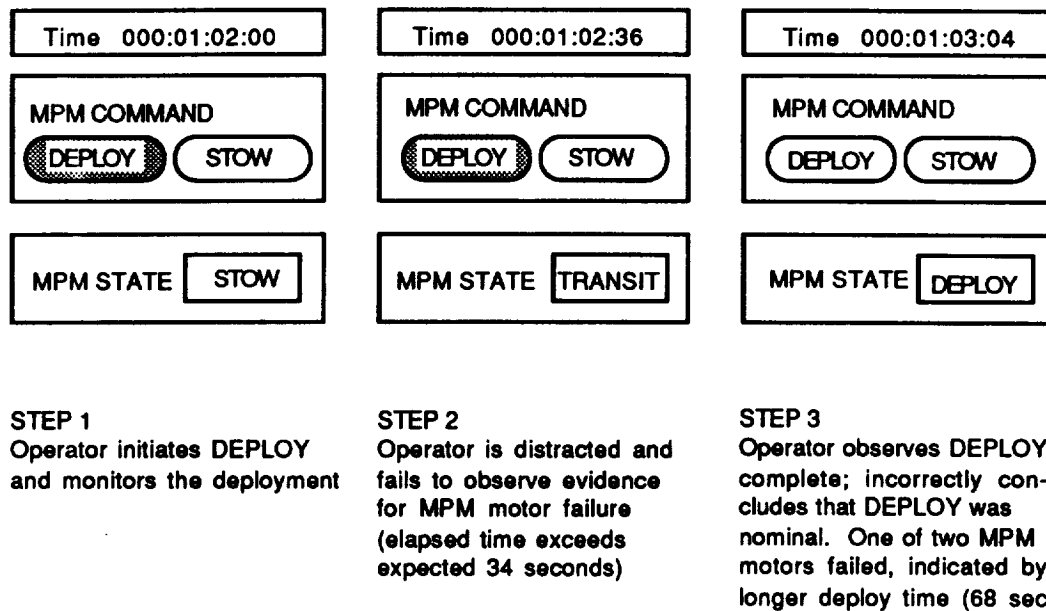


Figure 4-2. BEFORE: Example Illustrating Inappropriate Task Allocation

Problem: Task Allocation with Task Sharing

AFTER - Example Illustrating Solution:

In this example, the operator is assisted by the intelligent system. The intelligent system performs vigilance tasks such as monitoring elapsed time and state changes and sends messages identifying important events to the timeline portion of the display. The timeline summarizes both events and the timing of events. The operator is freed from closely monitoring time and state and now refers to the timeline of messages to identify important events occurring during the deployment. Notice that the operator can still access time and state information if needed, but is no longer required to watch them for transient values of interest.

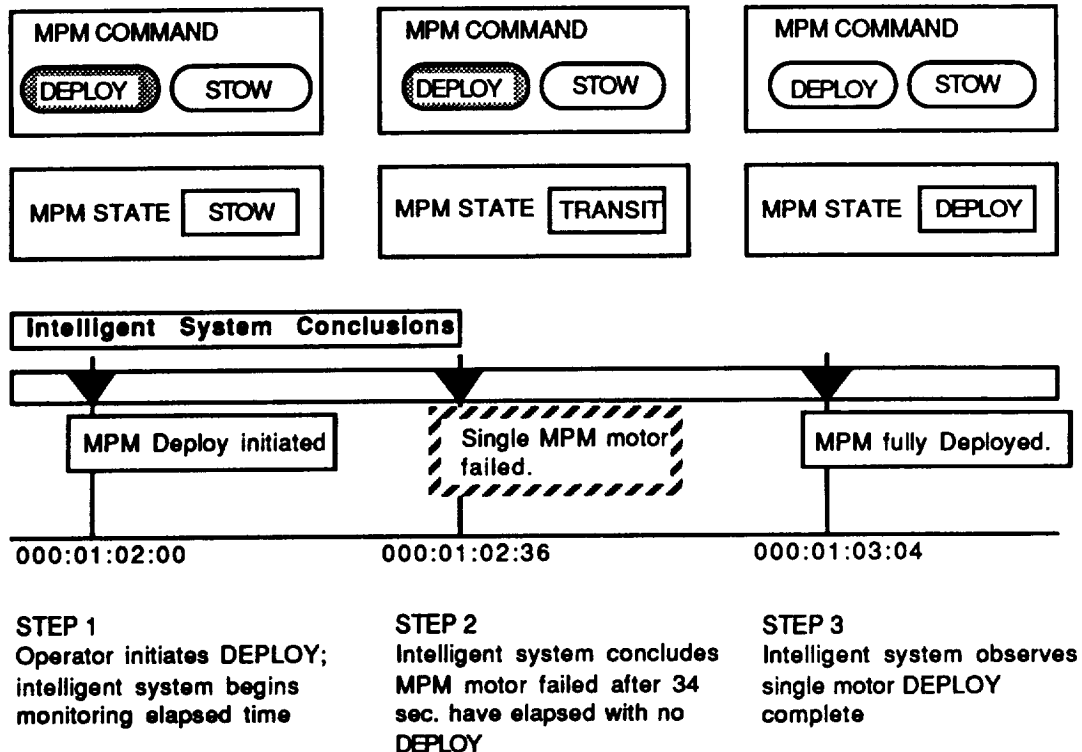


Figure 4-3. AFTER: Example Illustrating Improved Task Allocation

Problem: Maintaining Operator Control of Monitored Process

Recommendation: The operator should be consulted in decisions with potential mission or safety impacts. The intelligent system and its user interface should provide the operator with sufficient information about its reasoning and its activities associated with the monitored process to make fault management decisions and to effect these decisions. This includes access to information from the monitored process independent of the intelligent system, since the operator may be required to take over from the intelligent system (see section 4.1.3 on operator take over).

Example: The example shown in figure 4-4 illustrates a situation where the operator is excluded from an important decision affecting the monitored process. In figure 4-5, the example is altered to allow the operator the ability to control the monitored process. The intelligent system no longer prevents the operator from re-commanding the deploy. This example is based on Scenario 2 (refer to beginning of section 4).

Dynamic Task Assignment

In the concept of a fault management team, the initiative to perform operational tasks is shared between the operator and the intelligent system. This shared initiative allows both the operator and the intelligent system to alter their task assignments depending upon the current goals of the team and the ability of each agent to achieve those goals (what Johns (1990) calls variable or dynamic task sharing). For example, tasks can be dynamically re-assigned to balance work load between agents or to compensate for a loss of agent capability (i.e., failure). Note that only the team leader (i.e., human operator) can initiate re-assignment of tasks in this concept of the fault management team.

The operator should be cognizant of the alternate task assignments that are supported by the intelligent system. Often, intelligent systems have multiple levels of automation (e.g., activity executed automatically by intelligent system versus recommended activities executed by the operator) (Hefley, 1991). Each level represents a different task assignment. The operator should understand the range of available task assignments, know conditions under which each task assignment is appropriate, and be able to alter task assignments as needed.

Dynamic task sharing results in a set of possible activities for each agent, not all of which are performed under normal circumstances. This set consists of the nominal activities as well as activities that can be performed in atypical or anomalous situations. Compositely, all agents and their activities represent a structure, or team architecture, that defines the range of allowable activities for each agent and specifies how these agents must interact to jointly accomplish tasks. It is necessary to specify this architecture of agents and activities to identify complete information requirements.

Problem: Maintaining Operator Control of Monitored Process

BEFORE - Example Illustrating Problem:

This example illustrates a situation where the operator is excluded from an important decision affecting the monitored process. The indication of deployment has not been received from the sensor detecting deployment. RMS operations cannot continue until deployment is indicated. Based on his flight experience and the visual view of the RMS position, the operator concludes that the RMS is actually deployed, but the sensor detecting a deployment has failed. He recognizes that a re-command of deployment should put the sensor into the correct configuration. The intelligent system, however, is unaware of this undocumented procedure. It concludes that there is no reason to re-command deploy and prevents the operator from doing so. Thus, the operator has lost control of the monitored process and is prevented from taking the correct action.

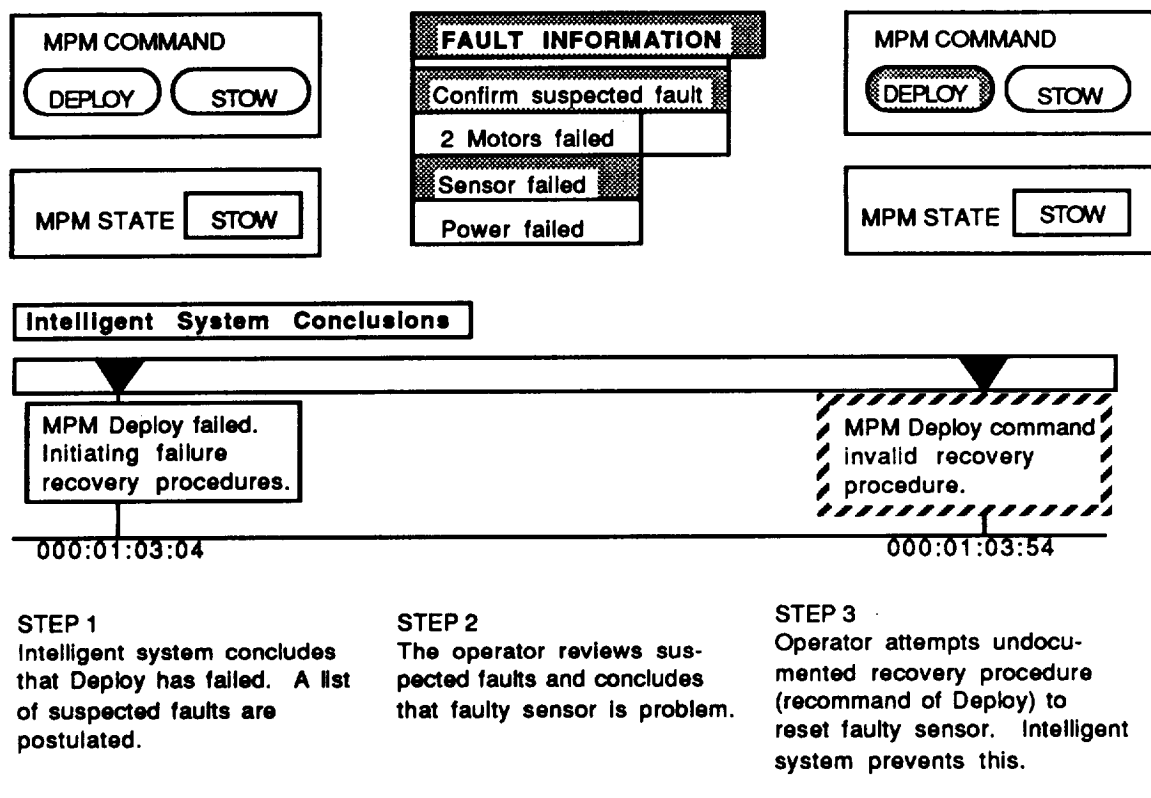


Figure 4-4. BEFORE: Example Illustrating the Loss of Operator Control over the Monitored Process

Problem: Maintaining Operator Control of Monitored Process

AFTER - Example Illustrating Solution:

In this example, the intelligent system no longer prevents the operator from re-commanding the deploy. It now issues a warning that MPM deploy is an invalid recovery procedure and provides the operator with an option to continue the re-command or abort the attempt. The operator chooses to continue and the intelligent system records that the operator overrode its recommendation. Since the appropriate corrective action has been taken, the deployment would now be completed.

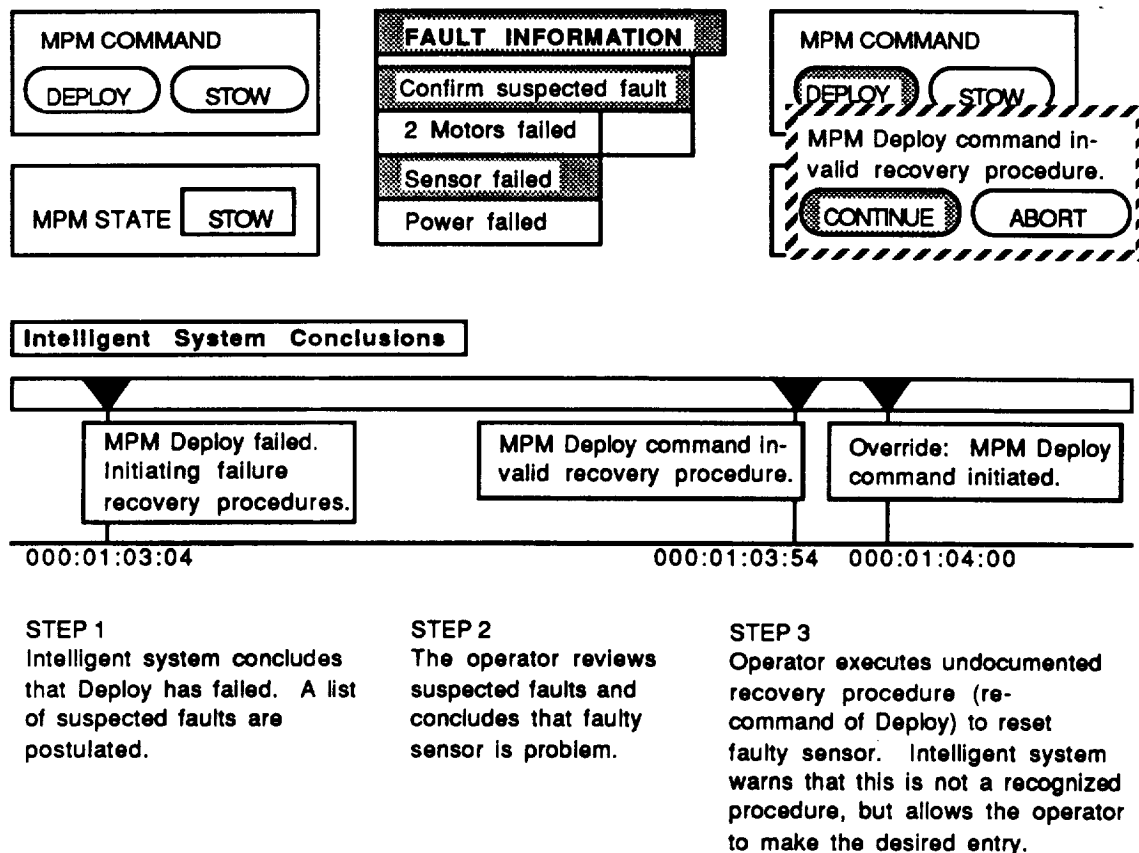


Figure 4-5. AFTER: Example Illustrating the Provision for Operator Control over the Monitored Process

Problem: Specifying the Architecture of Agents and Activities

Recommendation: Define the expected agent task assignments, corresponding agent activities, and ways that agents will interact to perform these activities. This specification of an architecture of agents and activities is essential in identifying the information required to achieve fault management goals and to coordinate joint agent activities.

These information requirements will affect the design of both the intelligent system (i.e., can it generate the required information?) and the user interface (i.e, does it provide access to the required information?).

The specification of an architecture of agents and activities should include activities in addition to those supporting normal domain tasks. Such activities include agent coordination activities required for the human and computer to work together effectively and contingency activities in response to unanticipated situations (section 4.2.3). Investigation of the interaction between humans when collaborating provides insight into human-computer coordination (section 3.3.2 and appendix C). Research is needed, however, into methods for identifying the complete range of agent activities.

Issue: Methods are needed to assist in defining this architecture of agents and activities, especially in identifying agent coordination activities for joint tasking.

In reality, it will not be possible to identify every possible way that the intelligent system and human will interact. Experience with teams of human operators has shown that unanticipated (and unplanned for) situations do arise during fault management and must be accommodated with new and unique activities. Unanticipated situations occur when either the monitored process or the intelligent system exhibits unexpected behavior. The tasks assigned to the operator and the intelligent system may be altered in such situations. The intelligent system should be designed to accommodate real-time alteration of normal activity sequences (i.e., the chronological series of activities performed by agents of the fault management team). See also section 4.2.3 for recommendations relevant to handling unanticipated situations.

Problem: Handling Alteration of Planned Activity Sequences

Recommendation: Unusual and possibly untried sequences of activities can be performed by team members during workaround to accommodate unanticipated situations. The intelligent system should be designed to allow changes in expected activity sequences.

Example: The example in figure 4-6 illustrates difficulties arising from the inability to alter activity sequences in real time. In figure 4-7, the operator is allowed to alter the activity sequence expected by the intelligent system. The intelligent system can now respond in a more useful fashion to on-going events. This example is based on Scenario 3 (refer to beginning of section 4).

Problem: Handling Alteration of Planned Activity Sequences

BEFORE - Example Illustrating Problem:

A power failure affecting the elbow pitch joint has resulted in loss of all but the BACKUP control mode. Normally, four control modes are possible. Malfunction procedures specify that the SINGLE mode would be used for the remaining joints (wrist and shoulder). The operator, however, decides to use DIRECT mode to ensure that the elbow joint position will remain fixed (i.e. the only restraint in SINGLE is gearbox friction) during the planned operation to open a sunshield on a deployed satellite. The intelligent system's expectation of SINGLE mode operations cannot be readjusted and its recommendations become irrelevant.

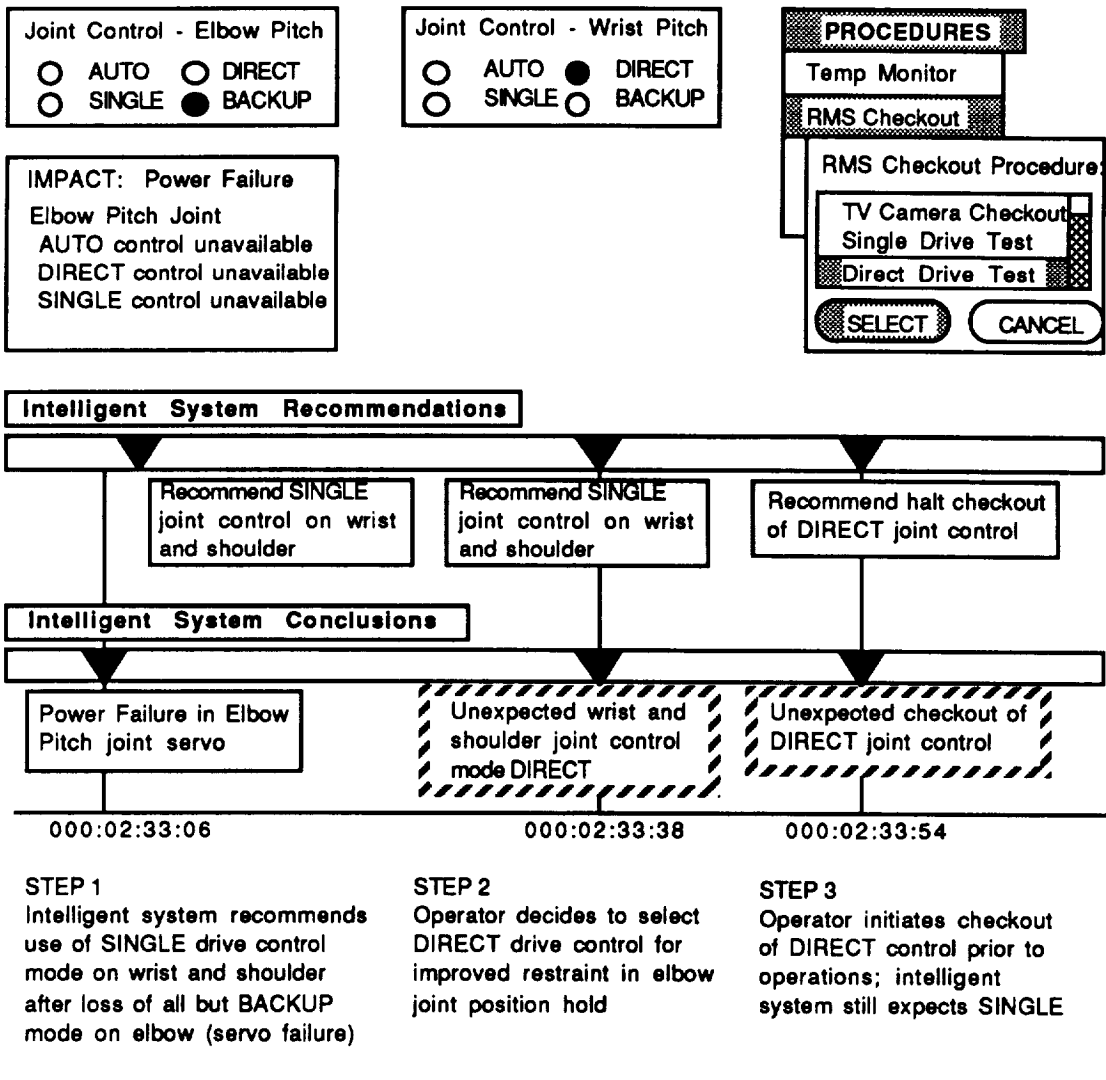


Figure 4-6. BEFORE: Example Illustrating Intelligent System with Fixed Planned Activity Sequence

Problem: Handling Alteration of Planned Activity Sequences

AFTER - Example Illustrating Solution:

In this example, the activity sequence expected by the intelligent system can be altered. The operator informs the intelligent system of the changes to planned activities. The intelligent system now assists the operator by assessing the potential impacts of the new sequence. Based on these impacts, the operator decides to continue with the new activity sequence and normal operations continue.

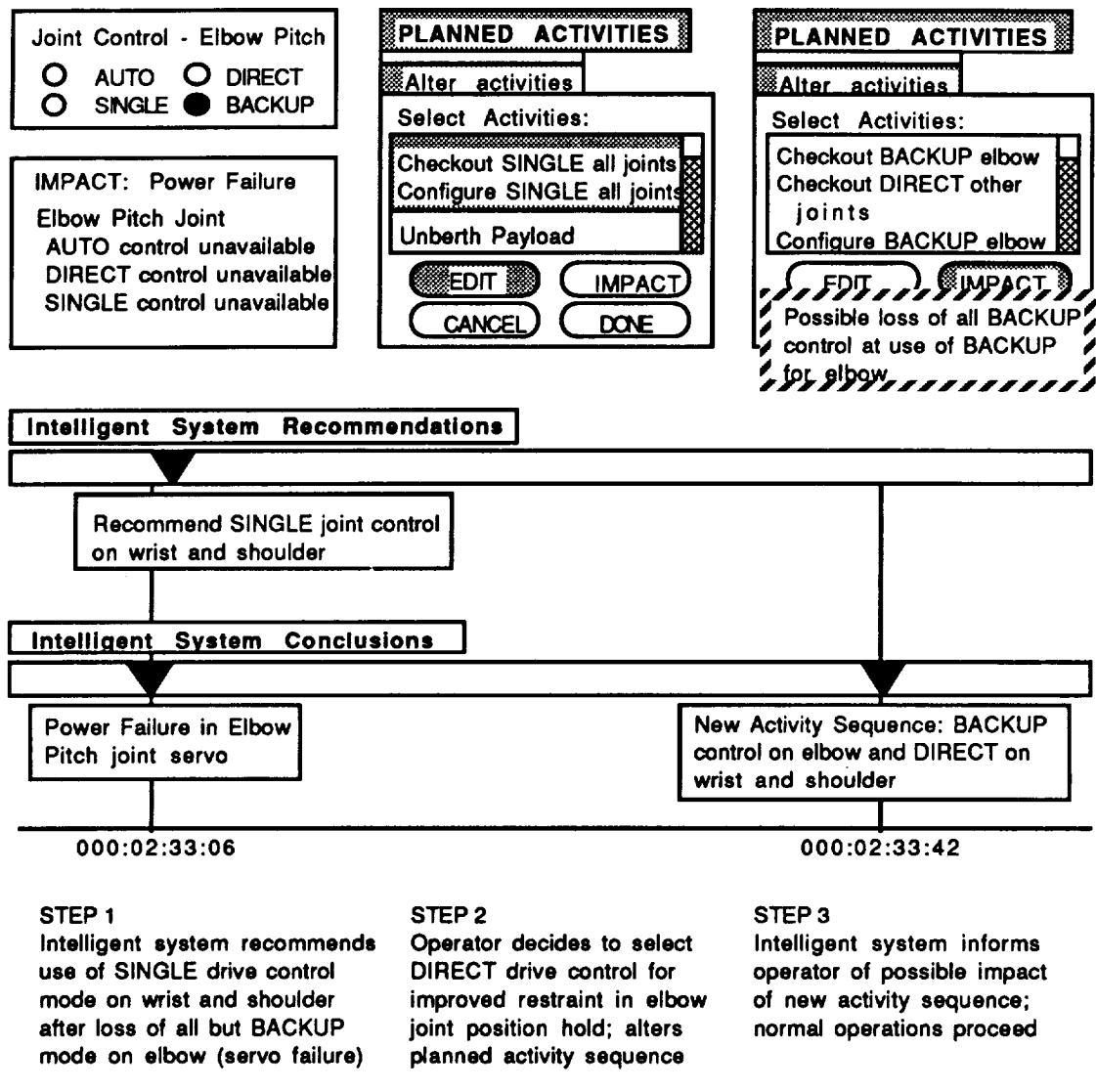


Figure 4-7. AFTER: Example Illustrating Intelligent System with Alteration of Planned Activity Sequence

Modes of operation are a way of varying agent task assignments in a controlled manner. Each mode represents an operational context that defines and constrains what the agent can do within that mode. Modes can also be a means of organizing and associating information for presentation based on goals associated with the operational context. Agent capabilities, the available information, and the style of agent interaction can vary in different modes. When modes are used to define contexts for agent activity and interaction, they should be clearly indicated to the operator at all times. Both the currently active mode and the remaining available modes should be evident to the operator. The display should be designed with a "uniquely different appearance" to clearly indicate the current mode (Johns, 1990).

Problem: Distinguishing Active Mode of Operation

Recommendation: When the intelligent system can operate in more than one mode, the user interface should clearly distinguish the currently active mode. The method used to identify mode should be easily discernable and should minimize the cognitive effort required by the operator.

Example: The example in figure 4-8 illustrates problems arising when the active mode of operation is not clearly distinguished on the user interface. In figure 4-9, active mode is indicated by a unique screen background pattern. This example did not require use of the scenarios outlined at the beginning of Section 4.

As illustrated in the example in figure 4-9, use of a distinctive background on displays relies on perception instead of cognition to indicate mode. Refer to user interface guidelines such as JSC (May 1988) or Smith and Mosier (1986) for a discussion of other coding techniques.

Example of systems from the case study with modes include the Procedures Interpreter (one of the OMA prototypes), the RTDS GNC Jet Control application, the Rendezvous Expert System, and the KU Band Self Test Expert System. See Volume 2 (Malin et al., 1991) for details of these cases.

Handover

Another situation where the task assignments of the operator and the intelligent system will change is at a handover of responsibility between agents. Handovers may occur between human operators at shift changes and between the human and intelligent system to balance work load or during joint tasking. The intelligent system should be designed to support both handover of task responsibility and to assist handovers between operators.

The discussion of agent coordination activities presented in section 3.3.2 includes a description of the ways in which agents can share a task. During supervised operations, the overseeing agent may temporarily assume responsibility from another agent (e.g., operator may switch to manual control of manipulator to avoid collision with intruding object). During sequential activity sequences, responsibility alternates between agents. For both types of operations, it is necessary for the agent assuming responsibility to be aware of relevant events preceding the handover to guarantee adequate understanding of the current situation.

Problem: Distinguishing Active Mode of Operation

BEFORE - Example Illustrating Problem:

In this example, the intelligent system provides a number of modes of operation. These modes affect the available information and functionality provided by the intelligent system. The operator enters the REVIEW mode, to allow viewing of previous events using the windows provided for REAL-TIME support. Notice that there is no visible indication that intelligent system mode has changed. Due to intervening work demands, the operator forgets that he is in REVIEW mode and attempts to perform an operation only permissible in REAL-TIME mode (alteration of activity sequence). The operator is reminded of current mode by a warning from the intelligent system that the requested activity is not valid during REVIEW.

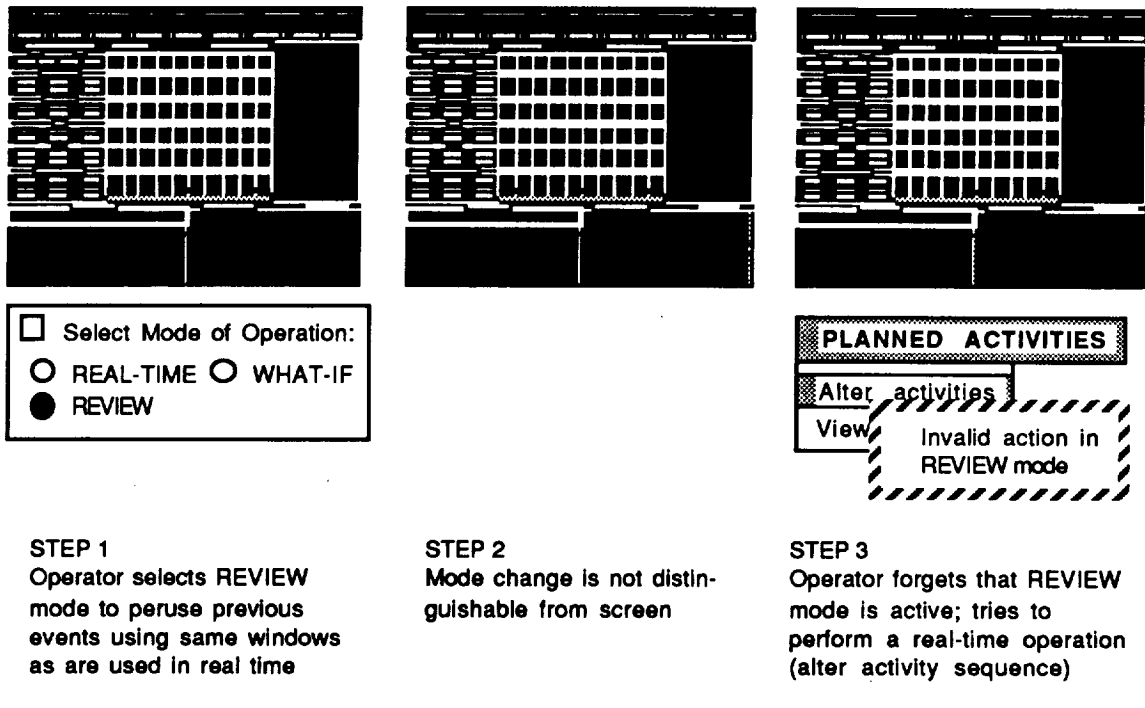
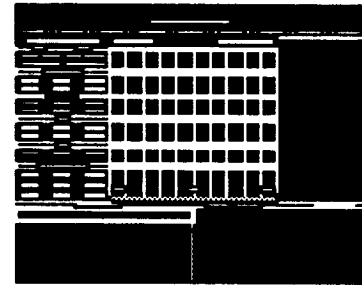
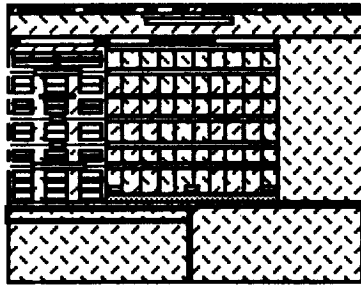
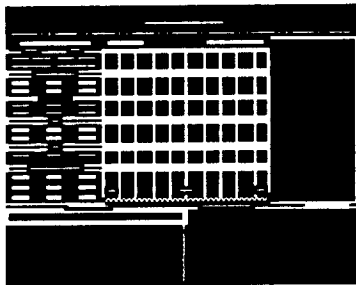


Figure 4-8. BEFORE: Example Illustrating Difficulty in Distinguishing Currently Active Mode of Operation

Problem: Distinguishing Active Mode of Operation

AFTER - Example Illustrating Solution:

In this example, active mode is indicated by a unique screen background pattern. A black background is used for REAL-TIME and a figured background is used for REVIEW. When the operator desires to change the activity sequence, he is immediately reminded of the active mode and makes the appropriate mode change prior to altering the activity sequence.



☐ Select Mode of Operation:

☐ REAL-TIME ☐ WHAT-IF

☒ REVIEW

☐ Select Mode of Operation:

☒ REAL-TIME ☐ WHAT-IF

☐ REVIEW

STEP 1
Operator selects REVIEW mode to peruse previous events using same windows as are used in real time

STEP 2
Mode change is obvious from change in screen background

STEP 3
Operator remembers to select REAL-TIME mode prior to performing real-time operation (alter activity sequence)

Figure 4-9. AFTER: Example Illustrating Clear Distinction of Currently Active Mode of Operation

Problem: Orienting In-coming Operator at Handover

Recommendation: Define and provide access to the information necessary to orient an agent assuming responsibility at handover about the current support situation. This information will include changes to scheduled activities and the baseline configuration of both the intelligent system and the monitored process as well as the current state and status of the monitored process and the on-going activities of other agents.

Example: The examples in figures 4-10 and 4-11 illustrate the need to orient in-coming operators at a handover about important events preceding his shift. In figure 4-10, the in-coming operator receives no information about these events. In figure 4-11, the intelligent system provides the in-coming operator with a summary of these events. This example is based on Scenario 3, with a handover from operator 1 to operator 2 inserted during the scenario.

During the development of the PDRS HCI design concepts, some simple features were provided for use by the in-coming operator at handover. An option to review all configuration changes to monitored process and intelligent system made by previous operators was provided. The description of these changes included useful annotation such as when the change was made, what the change was, who made the change, and why the change was made. This mechanism allows the operator to quickly orient himself on the current configuration of the monitored process and the intelligent system and provides a pointer for additional information by identifying the operator who made the change. The in-coming operator can then review the detailed activity logs of that operator or even personally discuss the changes.

To construct such a summary of events, the intelligent system must have access to information about important system configuration changes. If this information is not available in electronic form, the operator must provide it to the intelligent system. The intelligent system should also be informed when handovers occur and who the in-coming operator is (to allow annotation of changes). See also section 4.1.2 for discussion of visibility into intelligent system and 4.2.2 for discussion of visibility into the monitored process.

Interruption of Operator

As the situation changes in a dynamic environment, the importance of scheduled activities may change. New task priorities may require an agent to discontinue an on-going activity in order to perform a different activity. Such interruptions of agent activity are common in multi-tasking support environments with multiple, interacting agents.

The variety of sources of information typical of the flight support environment can result in situations where conflicting demands for operator attention occur. The potential for operator interruption is enhanced by the richer and more frequent dialogue and the more variable allocation of tasks resulting from the shared responsibilities and distributed tasking of the human-computer fault management team. The usual approach is to interrupt the operator with a new activity request and let him decide if the request should be pursued (i.e., provide the operator all of the information all of the time). An alternate approach is for the intelligent system to assist the operator in managing interruptions.

Problem: Orienting In-coming Operator at Handover

BEFORE - Example Illustrating Problem:

In this example, a handover from operator 1 to operator 2 occurs. Operator 1 fails to inform operator 2 of the change in activity sequence that was performed a few hours ago. This change affects the joint control mode used during RMS operations. When operations are initiated later, operator 2 erroneously attempts to select the nominal control mode (AUTO). He is then informed that another control mode is expected (BACKUP). Operator 2 must review logs from the past two hours to isolate the reason for this control mode change.

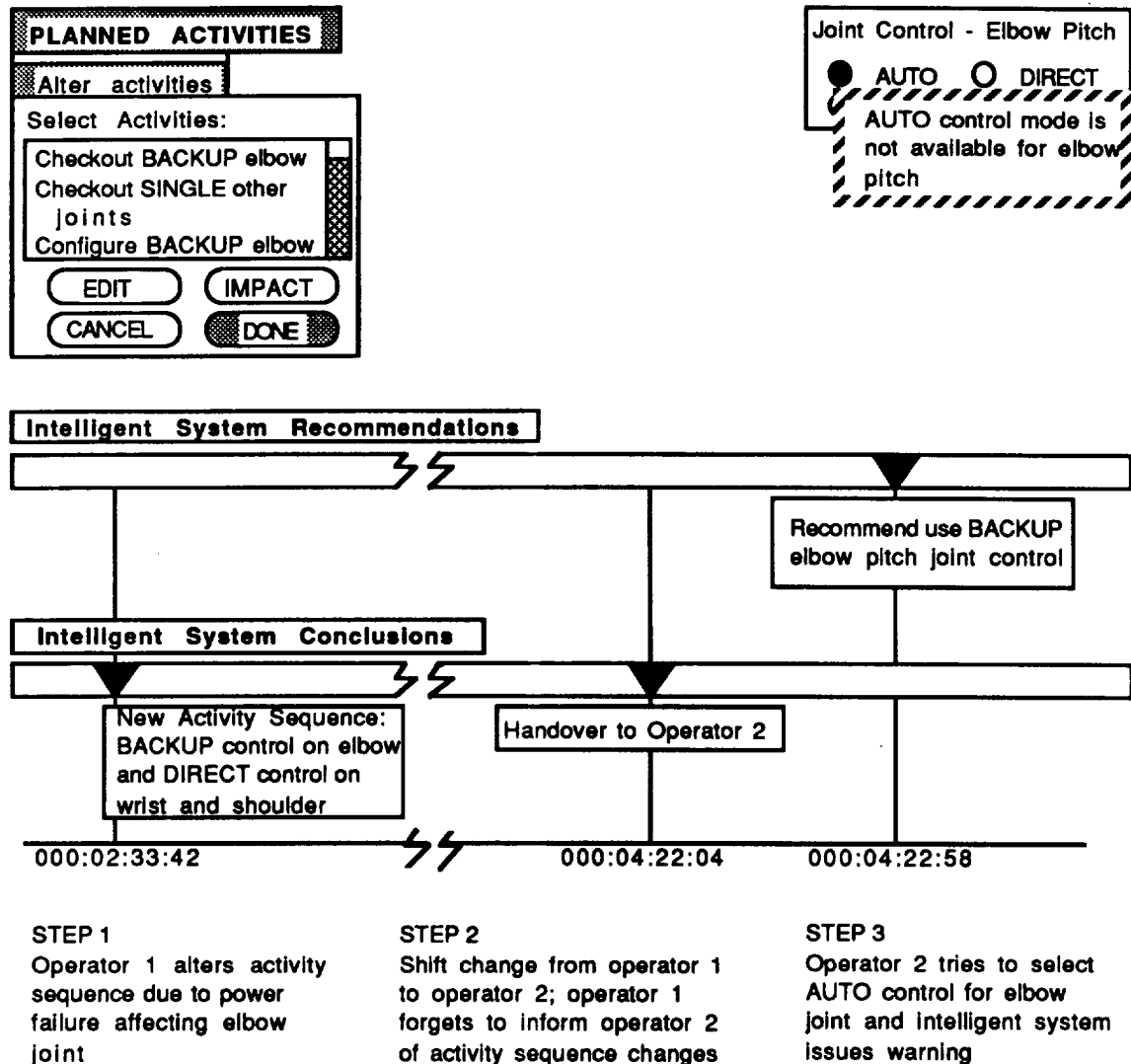


Figure 4-10. BEFORE: Example Illustrating Need to Orient Incoming Operators at Handover

Problem: Orienting In-coming Operator at Handover

AFTER - Example Illustrating Solution:

In this example, configuration changes made in previous shifts are made accessible to operators at handover. Operator 2 procedurally reviews these changes at the beginning of his shift. He thus becomes quickly familiar with events preceding his support period and responds correctly to the off-nominal control mode selected for RMS operations. Note that this type of capability requires that the intelligent system have access to information about these configuration changes, including when the change was made, what the change was, who made the change (i.e., identify of current operator), and why the change was made (e.g., operator annotation).

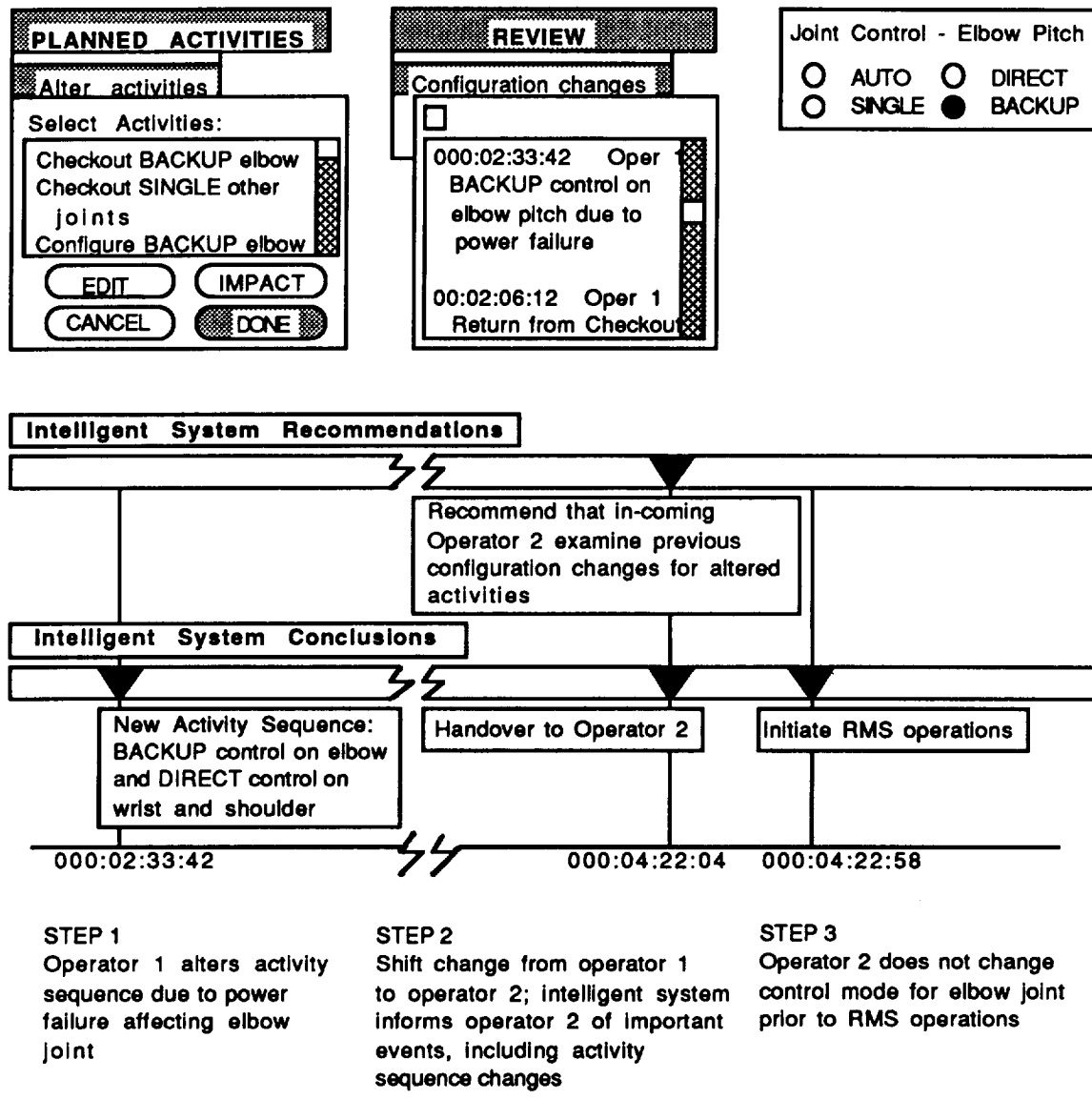


Figure 4-11. AFTER: Example Illustrating Intelligent System Orienting an Incoming Operator at Handover

Interruptions can initiate from a variety of sources and for a variety of reasons. The operator can decide to stop the current activity in favour of another activity deemed more important. The intelligent system can interrupt the operator with new information or activity requests. Other support personnel can desire to collaborate with the operator. Events in the flight support environment independent of the flight support task can interrupt on-going activities (e.g., failure of a workstation). Situations associated with the fault management task can interrupt the operator, including the occurrence of an anomaly, the completion of a background task requested by the operator, or a contingency situation (section 4.2.3).

There are two issues associated with managing interruptions of the operator. One issue is deciding which activity the operator should pursue based on the potential impacts of discontinuing the current activity or ignoring the new activity. The other issue is orienting the operator about the suspended activity when it is resumed after an interruption (either the former activity or the new activity).

Problem: Handling Interruptions and Suspended Activity

Recommendation: Agent coordination activities should include specification of how interruptions will be handled. New information and activity requests should be presented in a context that assists the operator in determining the impacts of discontinuing the current activity or ignoring the new activity until later. A means of suspending an operator activity while another is pursued should be provided. Information to orient the operator about a suspended activity when it is resumed should be provided.

Example: In the example in figure 4-12, nominal activities are interrupted by an anomaly. No capability has been provided to manage interruptions and the operator is required to do significant rework to resume to nominal activity at a later time. In figure 4-13, the operator is provided the capability to suspend and restore activities, allowing a quick resumption of nominal activities after the anomaly is addressed.¹ This example is based on Scenario 3 described at the beginning of section 4.

In Johns (1990), some techniques for managing the presentation of new activity requests are discussed. An icon can be used to indicate that an activity is waiting for the operator's attention. This suspended activity can be the request for a new operator activity (e.g., the review of new information) or a former activity waiting to be resumed.

Issue: Research is needed into ways to effectively manage the presentation of new information and activity requests to the operator when other activities are on-going. This issue is especially important in real-time, multi-tasking environments where multiple sources of information exist and requests are often time-dependent.

¹ Note that the capability to suspend activities introduces design challenges. The effects of a later activity could alter conditions required for a suspended activity or suspended activities could accumulate as task priorities change during an anomaly. Possible solutions to these challenges could include requiring the operator to verify conditions prior to activity resumption, limiting the number of activities that can be queued in suspension, and preventing the resumption of very old activities (i.e., activities that are more likely to have invalid conditions).

Problem: Handling Interruptions and Suspended Activity

BEFORE - Example Illustrating Problem:

In this example, the operator is interrupted during a nominal activity to manage an anomaly that has just occurred. The nominal activity is the alteration of an activity sequence. The operator is editing this sequence when the intelligent system detects an anomaly in the end effector. This anomaly requires execution of the End Effector Checkout procedures. The operator cancels the nominal activity and initiates activities to diagnose the failure. Much later, after the failure has been diagnosed and repaired, the operator desires to return to his previous activity of altering activity sequence. He must completely reconfigure the screen for this activity and then redo all editing changes.

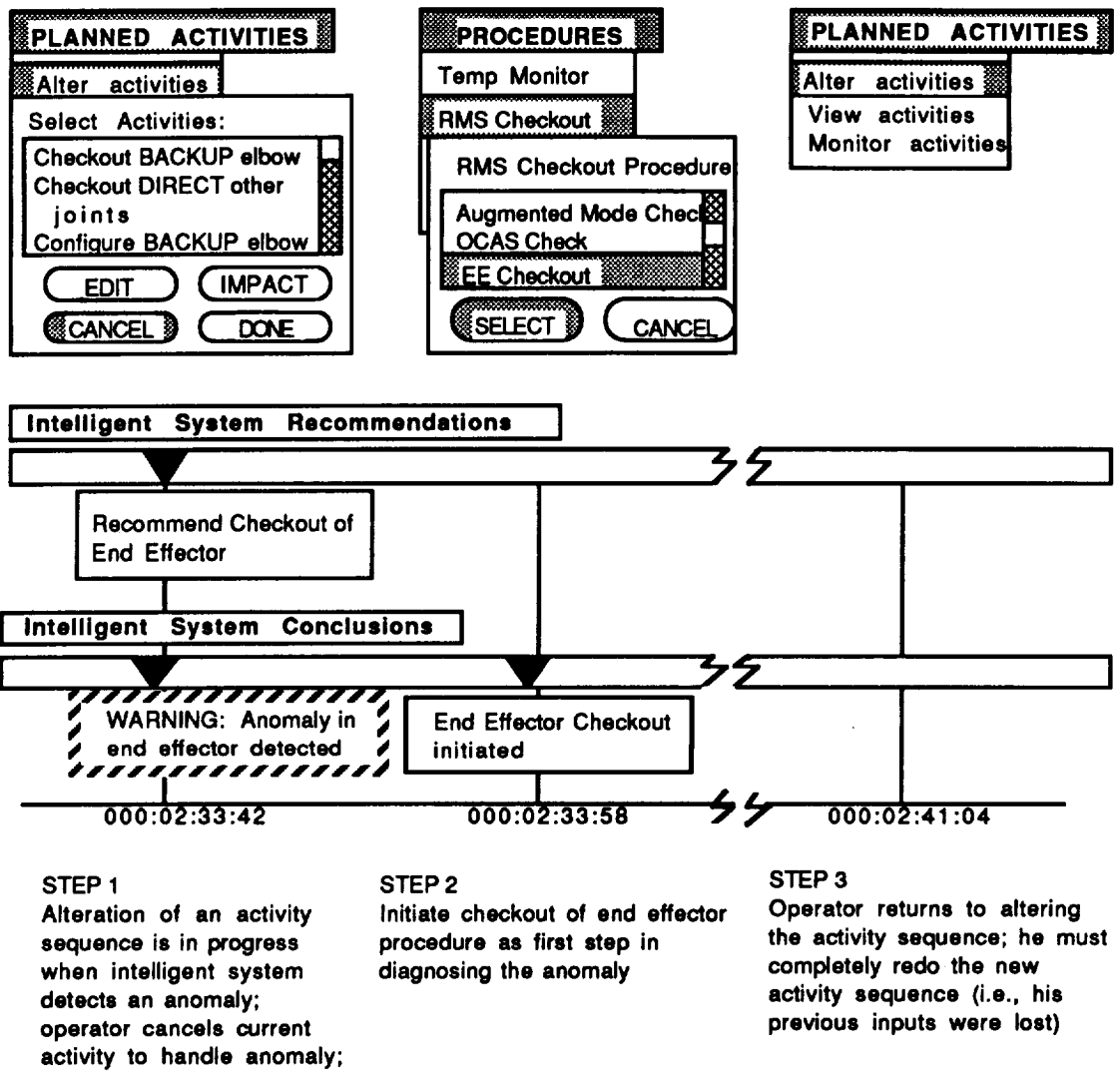


Figure 4-12. BEFORE: Example Illustrating No Capability to Handle Interruptions

Problem: Handling Interruptions and Suspended Activity

AFTER - Example Illustrating Solution:

In this example, the operator is provided the capability to suspend and restore activities. Suspend performs a temporary save on the current state of the activity. The operator suspends the activity to alter planned activity sequence before initiating diagnosis of end effector anomaly. The operator then restores the suspended activity after the anomaly has been diagnosed and repaired.

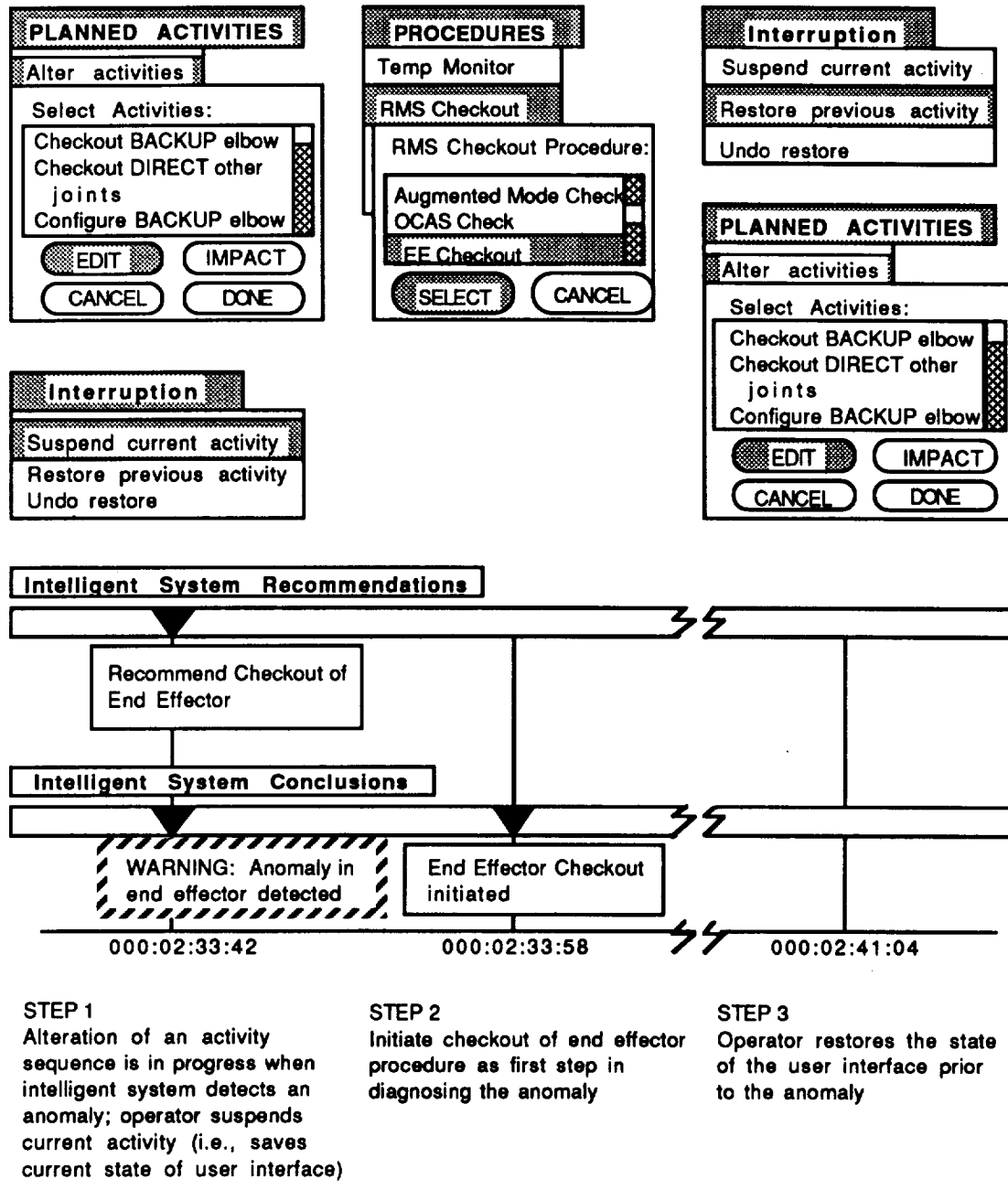


Figure 4-13. AFTER: Example Illustrating Assistance in Handling Interruptions

Techniques for handling interruptions that were observed in the case study include the use of icons indicating that new information is available for review (see Volume 2 (Malin et al, 1991), the DATA COMM Expert System) and the coding of messages to indicate the priority or importance of a message (see table 4-2 for an example from case study).

Table 4-2. Example of Coding for Message Priority (Brown and Kalvelage, 1988)

CLASS	OPERATOR RESPONSE	CODING
1	Immediate attention	Red, long tone
2	Attention as soon as possible	Yellow, short tone
3	Attention as soon as practical	Blue, short tone
4	Advisory, possible problem	Blue, short tone
5	Information, no problem	Blue, short tone

4.1.2 Collaboration between Agents

Collaboration between agents is essential to the coordination of agents for joint tasking. The goal of this collaboration is to establish a common understanding between agents of the team concerning some situation or event. This shared frame of reference assists in the interpretation of information provided by and activities performed by these agents.

A common understanding between the operator and the intelligent system is achieved by an exchange of information between the two about the environmental situation, the state of the monitored process, the beliefs of the intelligent system, and the beliefs of the operator. A human analogy to this is communication between partial, overlapping specialists, where the role of expert shifts depending upon the topic under discussion (section 3.3.2). In this scenario, both the intelligent system and the operator can be enlightened and their joint goal is to correctly perceive and respond to the current situation. The key issues in designing for collaboration are:

- What information should each agent provide?
- How may this information be effectively presented?

This section provides recommendations on the design of intelligent systems for collaboration. Traditional forms of explanation are investigated as a means of collaboration. Design of intelligent systems with access to system reasoning is presented as a way to improve operator understanding of system conclusions. Review of previous events is discussed as an approach for developing a shared view of situation between agents. Joint review of previous events leads to a common interpretation of these events. Joint prediction of the consequences of an event (section 4.2.2) can also establish shared expectations between agents and assist in identifying appropriate agent activities.

Visibility into Intelligent System Activities and Reasoning

Visibility can be described as providing an unobstructed view of an item of interest. Thus, providing visibility into intelligent system processing means providing the operator with access to information which characterizes the system processing, including internal or intermediate

states of the system. The operator should have access to the same information as the intelligent system to enable him to reach the same conclusions. Even the simplest form of explanation, the rule trace, requires some access to intelligent system internals. The ability to clarify intelligent system reasoning and actions becomes especially important when the system fails. Such a capability can be very useful in detecting software errors (section 4.1.3), both for debugging during software development (Chandrasekaran et al., 1989) and during operational support.

Problem: Providing Visibility into Intelligent System Reasoning

Recommendation: Visibility into the intermediate intelligent system hypotheses, alternate solutions, and internal states should be provided as a means of clarifying intelligent system processing/reasoning.

Example: In the example in figure 4-14, the operator is provided no visibility into the intelligent system. He has no ability to evaluate if the intelligent system is pursuing a productive path of investigation. In figure 4-15, access to interim information is provided, which clarifies the reasoning paths being pursued by the intelligent system. This example is based on Scenario 2.

Visibility into the intelligent system can include the display of interim information, including intermediate conclusions, hypotheses, and alternative solutions when available. Interim information is often conditioned on dynamic data and is thus highly variable. The display of such information should dynamically update as the situation changes (as represented by data). As illustrated in figure 4-15, interim information (such as hypotheses) should be clearly distinguished from facts.

Problem: Distinguishing Hypotheses from Facts

Recommendation: Providing access to interim results requires that hypotheses be distinguished from facts. The user interface should clearly identify the degree of certainty associated with displayed information to differentiate between the established and the inconclusive.

There are a number of coding methods that can be useful in discerning information items (e.g., methods identified in Herrman (1988) include color, intensity, reverse video, blinking, font, text size, and bolding). Symbols can also be useful in uniquely identifying characteristics of an information item. Johns introduces the *context icon*, a symbol displayed in proximity with a display item that provides the context for interpreting this item (Johns, 1990). Refer to the OMA Prototypes in Volume 2 (Malin et al., 1991) for an example from the case study that included access to intermediate information (e.g., list of suspected faults) from the intelligent system.

Problem: Providing Visibility Into Intelligent System Reasoning

BEFORE - Example Illustrating Problem:

In this example, the operator must wait until a message is issued to determine what the intelligent system is investigating. Even after messages have been issued, it is not clear how the system reached a conclusion and what hypotheses were eliminated. The intelligent system becomes an inscrutable "black box" to the operator.

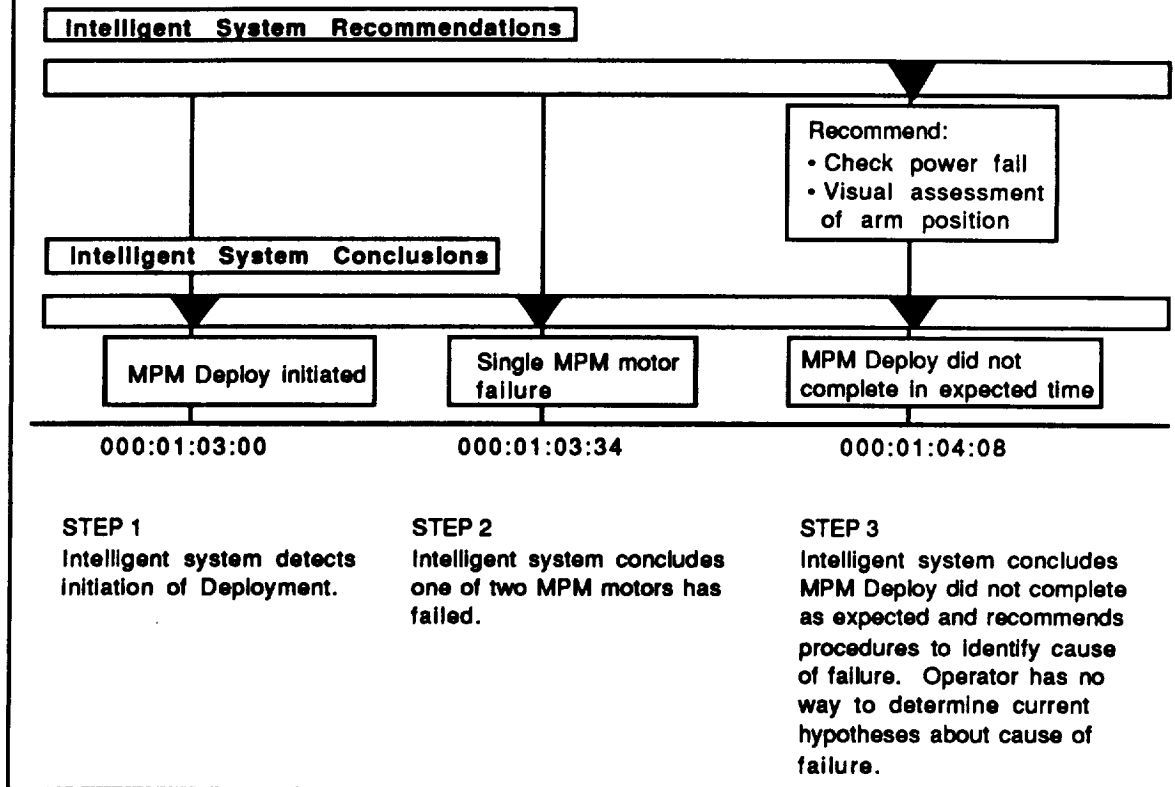


Figure 4-14. BEFORE: Example Illustrating No Visibility into Intelligent System's Reasoning

Problem: Providing Visibility Into Intelligent System Reasoning

AFTER - Example Illustrating Solution:

The operator is provided access to interim hypotheses in this example. This visibility improves operator understanding of the intelligent system's reasoning and also provides a possible means of altering information used in that reasoning (see also section 4.1.3).

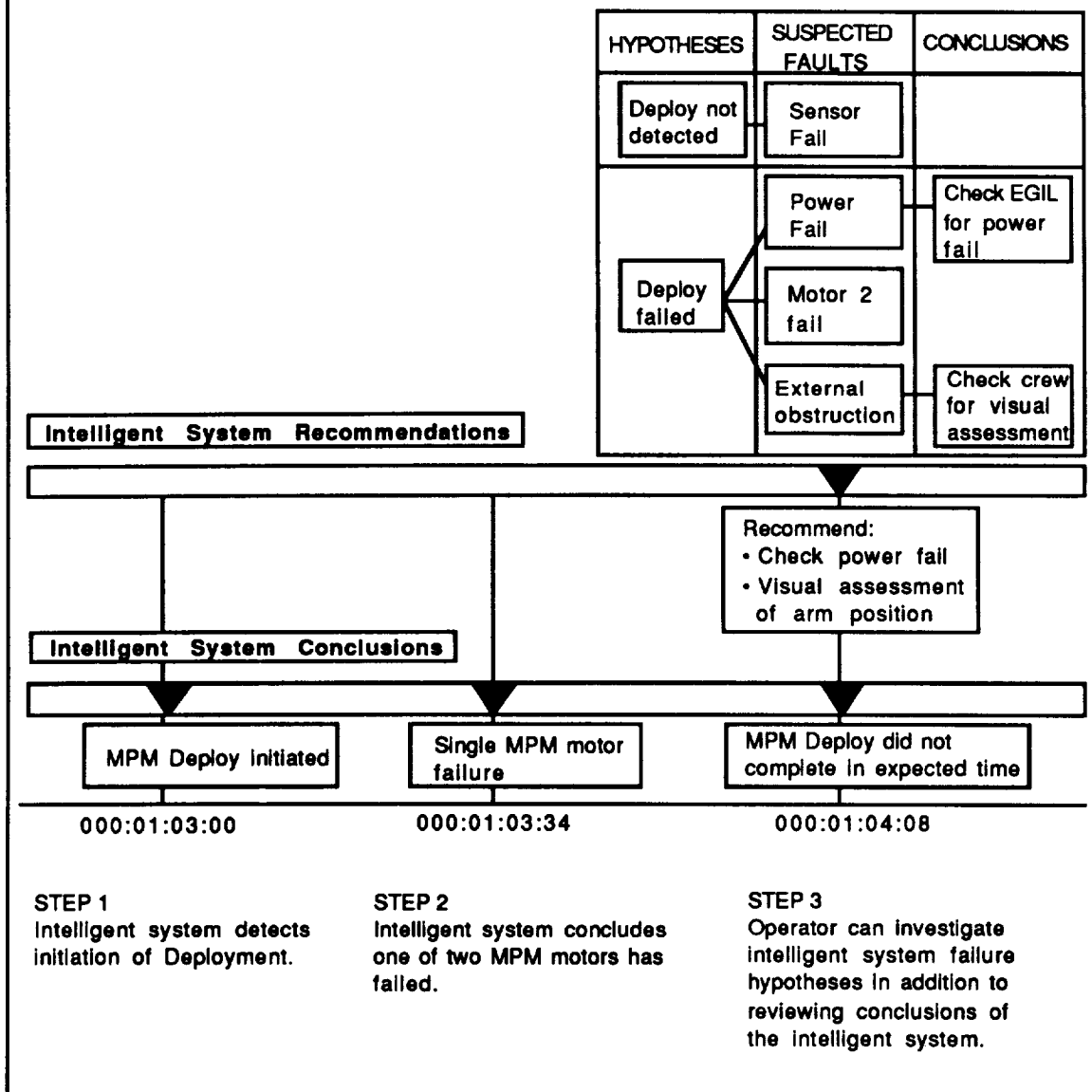


Figure 4-15. AFTER: Example Illustrating Visibility into Intelligent Systems Reasoning

Providing access to information and presentation of that information during collaboration will affect the intelligent system architecture and knowledge representation. The intelligent system must be able to provide information about its goals, hypotheses, interim states, alternative conclusions, and reasoning processes as well as the state of the monitored process. The knowledge base must be structured to provide this information in a usable form. The information required for agent collaboration should be defined as part of the initial system design. The internal knowledge representation and reasoning structure of the intelligent system should be accessible by the operator and should be used in communicating with the operator. Such a mental model assists the operator in evaluating intelligent system conclusions.

Problem: Understanding Intelligent System Reasoning Strategy

Recommendation: A representation of the reasoning strategies used by the intelligent system is an effective means of presenting information during collaboration between the operator and the intelligent system about the results of that reasoning.

Example: The example in figure 4-16 illustrates difficulties arising from no access to intelligent system reasoning strategy in real time. In figure 4-17, an explicit representation of the process for detecting anomalies is used to clarify this strategy. This example is based on Scenario 2.

Multiple levels of automation in the intelligent system can result in variations in the reasoning process of the intelligent system. Visibility into the intelligent system reasoning strategy should include clarification of the available levels of automation when more than one level is provided. The representation of this reasoning strategy should assist the operator in understanding and using these levels of automation. See also the discussion of dynamic task assignment in section 4.1.1.

As seen in figure 4-17, a functional diagram can be used to represent the reasoning strategy used by an intelligent system. This functional diagram can be explicitly represented in the user interface. This explicit representation reinforces the operator's understanding of the intelligent system's capabilities and can serve as a means of accessing interim conclusions and alternative states used in the reasoning process.

The use of functional diagrams for display can introduce the problem of insufficient display space, especially if the diagram is complex (Czerwinski, 1990). One method for managing the use of display space by the diagram is to partition it into major functional blocks and employ optional overlay for investigation of the details of a functional block (Chu, 1990). When the use of overlay results in only a portion of the flow chart being visible, an overview of the entire flowchart should be displayed, with the currently visible portion highlighted (see also section 4.3.2 and section 5). These techniques were used in the PDRS HCI design concepts (Schreckenghost, 1990).

Problem: Understanding Intelligent System Reasoning Strategy

BEFORE - Example Illustrating Problem:

In this example, the operator does not understand why the intelligent system has decided to issue an alarm at this time. He recognizes that the MPM is not in the desired state, but is not clear about the timing of the alarm. By reviewing the history of messages issued by the intelligent system, the operator infers that timing criteria are exceeded, but it still not clear what those criteria are. He finally has to resort to looking the information up in a manual and assuming that the intelligent system is using the documented criteria.

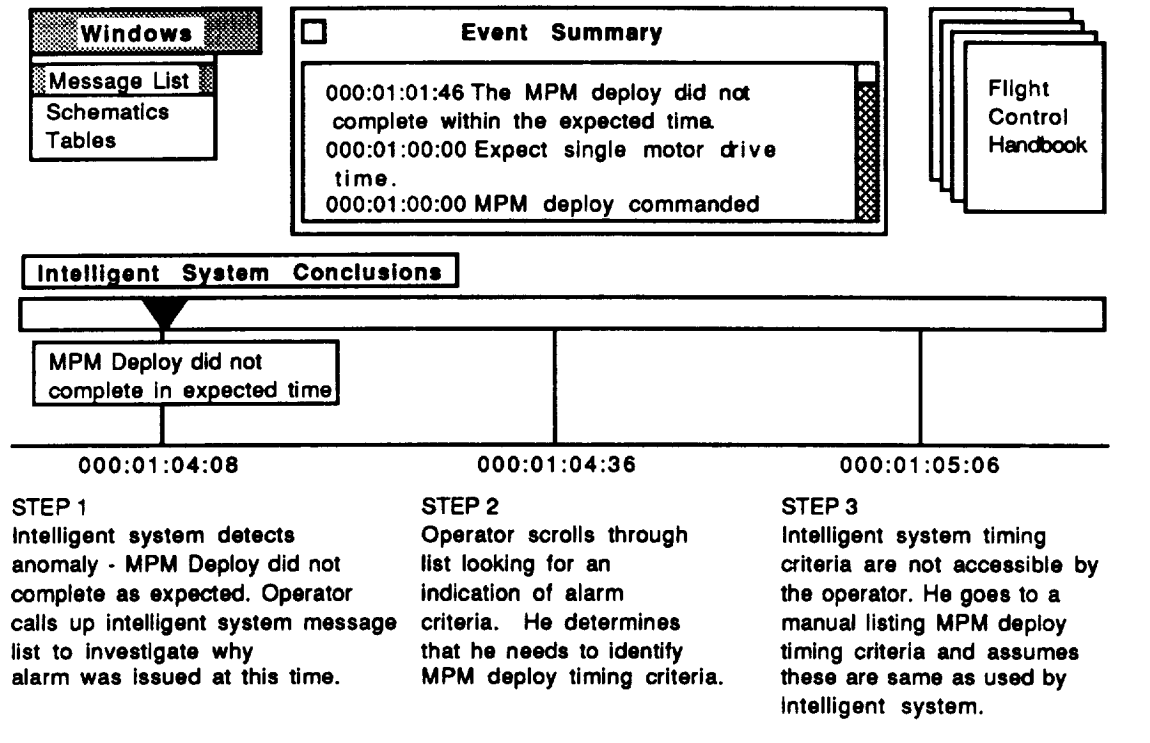


Figure 4-16. BEFORE: Example Illustrating Inability to Access Intelligent System Reasoning Strategy

Problem: Understanding Intelligent System Reasoning Strategy

AFTER - Example Illustrating Solution:

In this example, diagrams illustrating portions of the reasoning process are used as a means of accessing information about the reasoning strategy. These diagrams provide a context for interpreting the information provided (i.e., how was this information concluded?). Additionally, these diagrams serve to reinforce the operator's understanding of intelligent system reasoning.

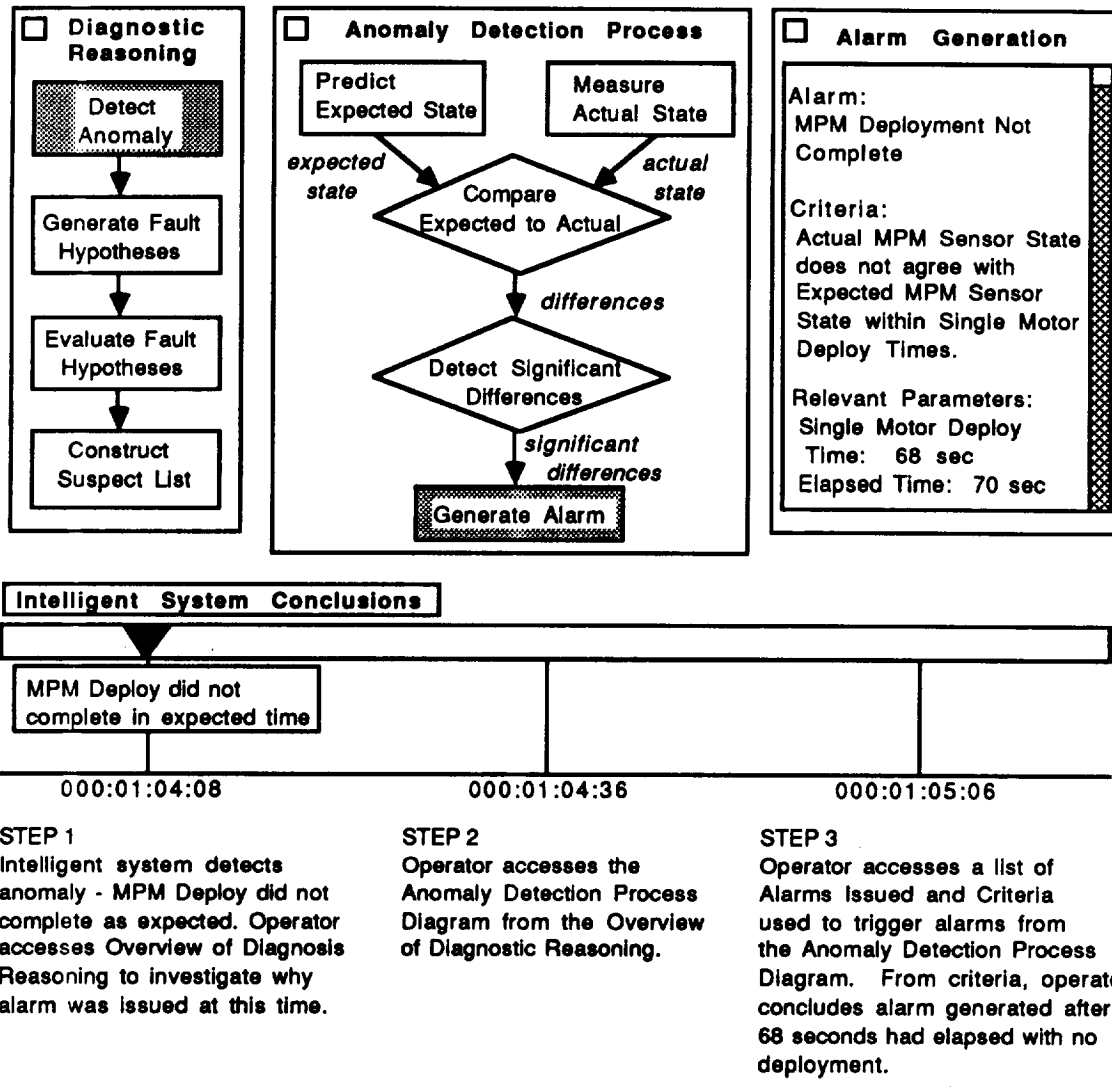


Figure 4-17. AFTER: Example Illustrating Explicit Representation of Intelligent System Reasoning Strategy

Explanation

Explanation is an iterative process of identifying, organizing, and presenting information to satisfy an agent's inquiry about some situation (or event). The on-going activities of the participating agents establish a context for the interpretation of information that will affect the response provided (Chandrasekaran et al., 1989). The currently-active goals and activities of the inquirer should be considered by the responding agent when formulating the answer to a question. This context should influence the presentation of information and the style of agent interaction. For example, when the intelligent system is monitoring data in real time, it should explain its conclusions based on the current situation. When the intelligent system is assisting in predicting the effects of some hypothetical event, it should explain its conclusions based on the hypothesized situation.

Problem: Responding to Questions in Context

Recommendation: If the intelligent system is to provide an explanation capability, it should be designed to interpret and respond to questions within the context of relevant situations and agent activities. This consideration affects both the information content and presentation of the answer.

Example: In figure 4-18, the example illustrates operator confusion resulting from no consideration of context by the intelligent system during collaboration. Figure 4-19 illustrates improved collaboration when the on-going activities are used to provide context. This example is based on Scenario 3.

Effective communication requires shared modalities and compatibility with information processing capabilities between the communicants. It is important for intelligent systems to include knowledge which is at least similar in important respects to the knowledge of their human partners. Research suggests that human experts use *mental models*, systematic inferential schemes grounded in their commonsense models, in reasoning about physical systems (Gentner and Stevens, 1983). In artificial intelligence, *qualitative physics* grew out of the attempt to formalize mental models and engineering problem solving. Qualitative physics seeks to formalize reasoning about the physical world, from the commonsense intuitions of the person on the street to the technical analyses performed by expert scientists and engineers (Weld and de Kleer, 1990).

Domain experts rely on a shared substrate of knowledge about the physical world. For example, human physicists have grown up in such a way to arm them with intuitive notions of space, time, matter, and so on, which provide a foundation for more technical knowledge gained in schools and laboratories. To be sure, this foundation is (ideally) rebuilt during the process of learning. Both experts and novices can often solve textbook problems, but one mark of a true expert is that their intuition has been systematically informed and upgraded by their technical education¹. But the point is that this shared experience provides the starting point for communication between domain experts (see section 3.3.2 and appendix C for discussion of modes of interaction between experts).

¹ Although the Clement and McCloskey chapters in (Gentner and Stevens, 1983) indicate that this conceptual restructuring does not always occur during physics education.

Problem: Responding to Questions In Context

BEFORE - Example Illustrating Problem:

In this example, nominal activities are interrupted by an anomaly requiring the operator's immediate attention. This transition in activities represents a context change. The operator is now focused on diagnosing the anomaly and communications about the previous activity can be confusing. In this case, two impact assessments have been requested. Since diagnosis of the anomaly is the operator's current activity, he is expecting to see impacts of the anomaly. When the impact of the activity sequence change (i.e., the interrupted activity) is provided instead, the operator misinterprets it as the anomaly impact and fails to take action to compensate for the impact.

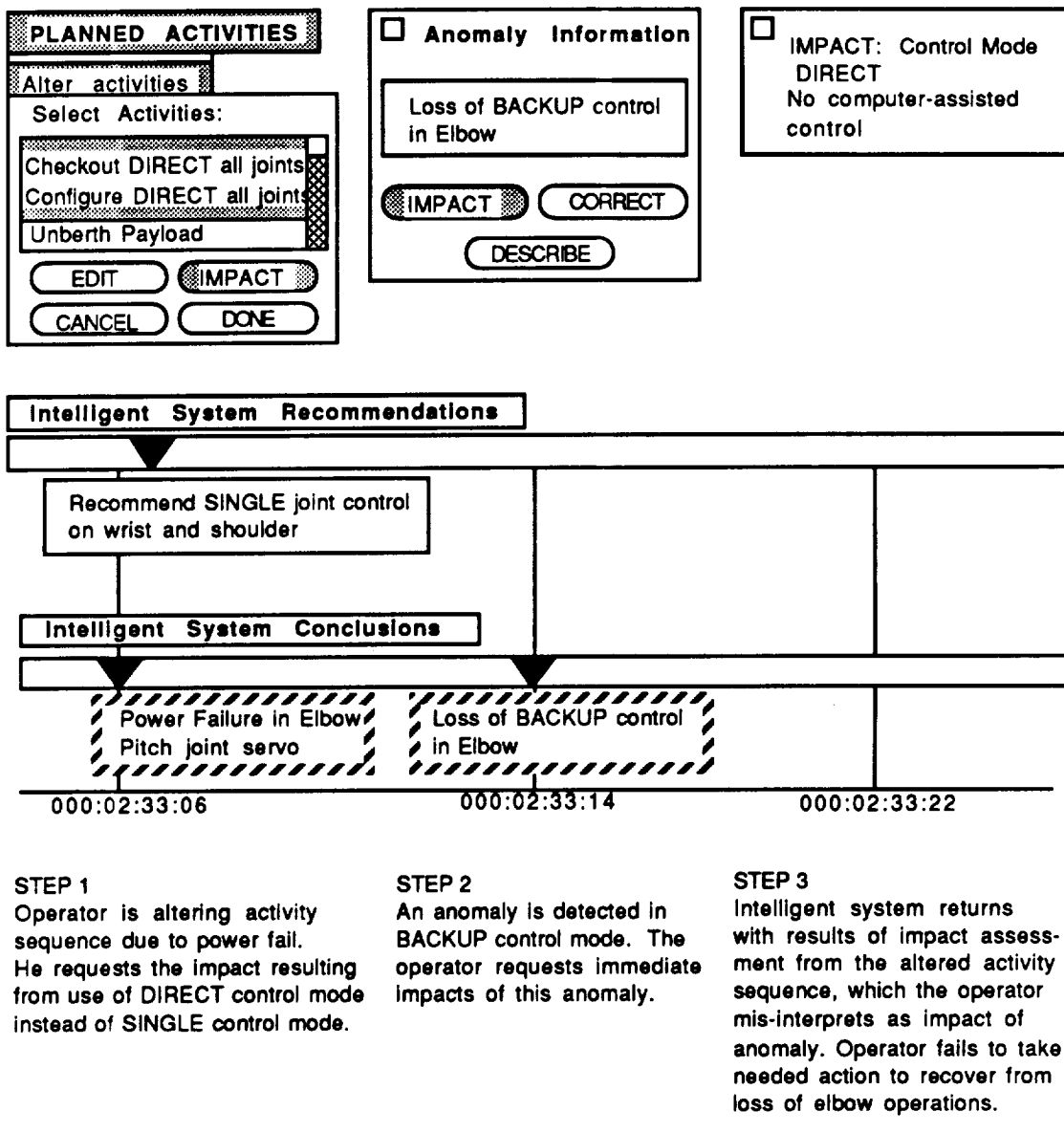


Figure 4-18. BEFORE: Example Illustrating No Consideration of Context during Collaboration

Problem: Responding to Questions In Context

AFTER - Example Illustrating Solution:

In this example, the intelligent system recognizes the transition to another activity and alters its activities correspondingly. The operator is provided with the expected anomaly impact assessment while the impact assessment for modifying activity sequence is suspended until the anomaly is under control. The operator then proceeds to compensate for the loss of elbow operations.

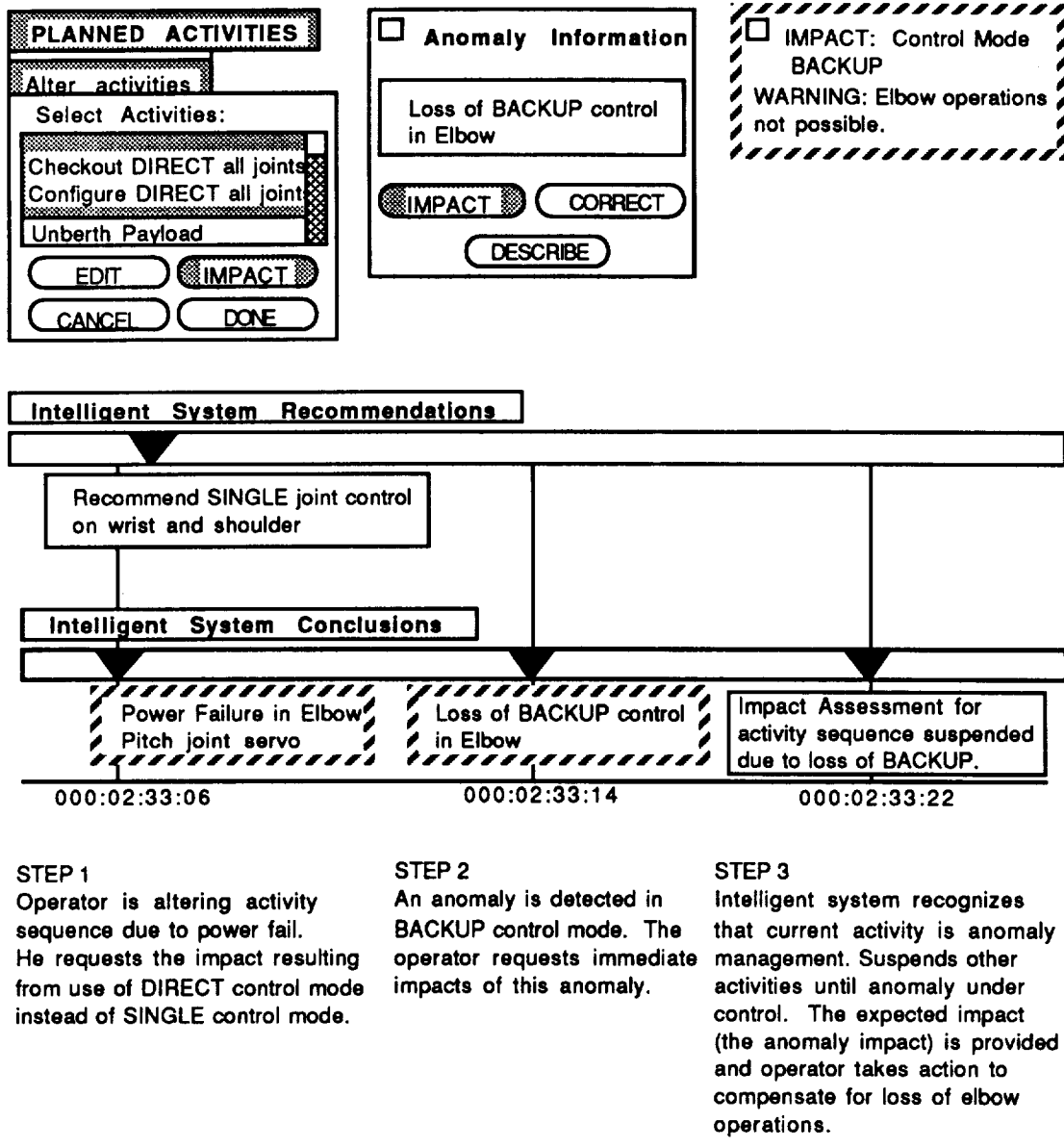


Figure 4-19. AFTER: Example Illustrating Improved Collaboration using Context of On-going Activities

Obviously, our computer programs have not had the benefit of this shared experience. And this puts them at a serious disadvantage, both in reasoning about physical systems and in communication regarding them. Since a general-purpose, human-grade intelligent system which could learn from living in the world seems a rather long way off, the only way to bridge this gap is for us to articulate this tacit knowledge of the physical world, in a precise form which our programs can then use. Using this concrete representation of intuitive knowledge, the operator and intelligent system can share knowledge about the monitored process that is normally gained by only through experience. The key point for human-computer interaction is that, just as the visual and auditory aspects of user interface must match the constraints of the human information processing system for maximum effectiveness, the information content of the user interface must also match the constraints of human conceptual systems if we are to ever make computers into collaborators.

Problem: Making Abstract Entities Concrete

Recommendation: A variety of abstract, conceptual entities are often used in reasoning about complex physical systems. Many of these concepts are gained through shared experience. The HCI design should make these abstract entities concrete. One obvious kind of conceptual entity needed to support communication is the existence of a specific physical process (e.g., chemical reaction, evaporation). Such information could be provided as annotations to the artifact's schematic, or used as the basis for a completely different display organization.

Annotated schematics seem to be the dominant display organization in today's model-based systems. While they clearly have an advantage in being familiar to domain experts, it is far from clear that they are optimal for most purposes. One alternative class is functional descriptions, which can provide valuable suppression of detail. Displaying the block currently attended to provided an excellent indication of the system's reasoning.

The operational goals of the system should be made explicit and displayed along with their connection to performance parameters of the system (the formalisms used in cognitive engineering may be useful in this regard). A good example of this was the "efficiency meter" in the Recovery Boiler Tutor (Woolf et al., 1986). No such meter existed in the actual plants, but it proved so useful in the evaluation of the tutor that the addition of such a meter was under consideration.

An alternative to organizing parameters visually using a schematic is to draw the presumed causal structure connecting them. Such a display could provide a very rapid indication of where to look to pinpoint faults and would be useful in a joint effort to verify a diagnosis.

A qualitative summary of the artifact's behavior, in the form of an event sequence, could be useful for many purposes. This low-level description could be continually parsed using a goal-oriented vocabulary to provide concise descriptions of what is going on. (For example, "it managed to compensate for the start-up of the semiconductor furnace already, but the temperature is still a bit high from the shutdown of the B evaporator."). See also section 4.3.1.2 on qualitative representation.

The ability to collaborate about the current situation requires that the operator understand both the environmental evidence (i.e., data) suggesting the situation and the knowledge and reasoning used by the intelligent system to draw conclusions about that evidence (i.e., shared knowledge between operator and intelligent system). Typically, explanation addresses

questions about the monitored process that generated environmental behavior. The operator also should be able to ask questions about the intelligent system decision-making process -- what reasoning strategies were used, what were the interim states that lead to a decision? Agents should be able to exchange information about (1) the environmental influences, (2) the behavior of the monitored process (including the control strategies that affected that behavior), (3) intelligent system beliefs, goals, and strategies, and (4) operator commands. This information should be presented within the context of what has preceded the current situation and why specific activities were performed. Such information exchange will assist the operator in evaluating the conclusions of the machine and aid in identifying ways to redirect intelligent system reasoning if required. If collaborative exchanges are logged, they can also be used off-line to refine the intelligent system's knowledge base.

Information sharing¹ should occur in both directions, where both the intelligent system and the operator can inform the other about its beliefs. Since the operator is the "team leader", however, his beliefs will predominate. Designing the intelligent system for improved collaboration should allow the intelligent system to influence the operator's beliefs more effectively.

Problem: Promoting Shared Understanding of Situation

Recommendation: Explanation should promote a shared understanding of situation by participating agents. It should assist the operator in identifying problems and their relation to task goals. It should include background about the perceived situation and describe on-going activities and events that affect this situation.

Example: The example in figure 4-20 illustrates problems that can arise when the intelligent system is not designed to assist the operator in understanding its conclusions. In figure 4-21, an example of a design for shared understanding is illustrated. This example is based on Scenario 2.

Techniques for explanation need not be limited to textual presentation. The example presented previously in figure 4-21 shows how a graphic illustration of function may be used as a type of explanation. Notice that the use of graphics for explanation does not preclude the use of text. Just as human-human explanation uses the spoken word to supplement graphical illustration, human-computer explanation using graphical displays can be improved by text annotation (Johns, 1990).

Explanation for real-time monitoring and fault management is affected by the time constraints of the support task. There is no time to wait until an anomalous situation has stabilized to conduct a retrospective dialog with the intelligent system (i.e., reconstruct what actually happened after the situation has fully developed). The intelligent system must provide assistance as the problem is unfolding, assisting the operator in formulating a response to the anomaly by clarifying goals, identifying impacts, and evaluating alternatives.

¹ Note that the concept of information sharing does not imply the need for user modeling in the intelligent system.

Problem: Promoting Shared Understanding of Situation

BEFORE - Example Illustrating Problem:

In this example, the operator is investigating why the intelligent system issued the alarm "MPM deployment not complete" at this time. The only means provided to assist the operator in understanding the reasoning of the intelligent system are a summary of event messages and a rule trace. The rule trace is a time-consuming approach, however, requiring an operator to become very familiar with implementation-level details of the system. Also, this approach does little to clarify the overall reasoning structure of the system.

☐ **Event Summary**

000:01:01:46 The MPM deploy did not complete within the expected time.
000:01:01:10 Expect single motor drive time.
000:01:00:36 MPM deploy commanded.
000:00:59:20 RMS Powerup complete.
000:00:55:04 RMS Powerup initiated.

STEP 1

The operator first reviews messages from the intelligent system by scrolling through the event summary. Although messages summarize important events, there is little indication of how these events were identified.

Rule Fired RMS1009-mpm-deploy-not-complete
Fact mpm_state stowed
Fact mpm_command deploy
Fact mpm_status single-motor-time-deploy
Fact current_time 000:01:01:46
Fact elapsed_time_deploy 70
Rule Fired RMS1108-mpm-single-motor-deploy
Fact mpm_state stowed
Fact mpm_command deploy
Fact mpm_status single-motor-time-deploy
Fact current_time 000:01:01:10
Fact elapsed_time_deploy 70
Rule Fired RMS1008-mpm-deploy-commanded

STEP 2

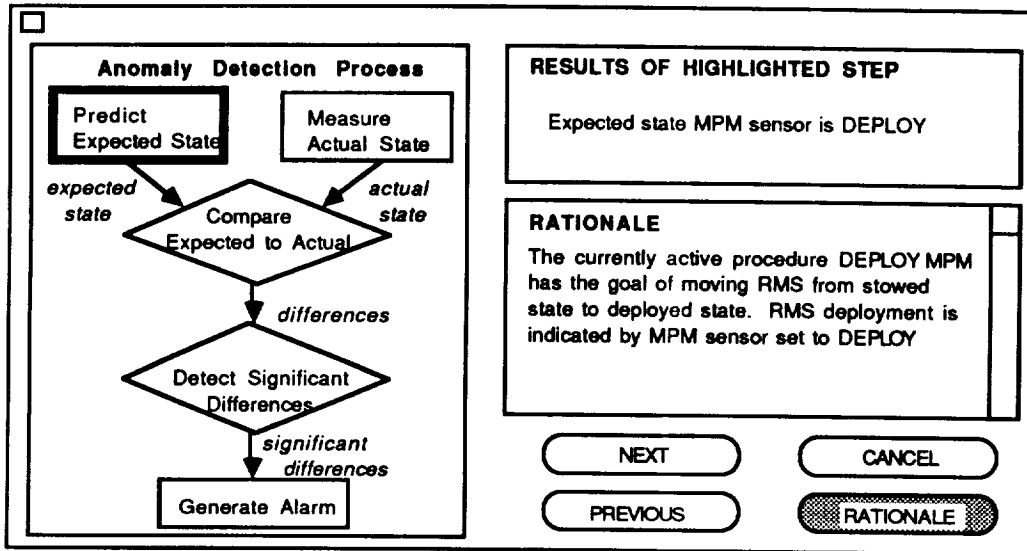
Next the operator calls up a rule trace which shows the rules that fired and the state of the knowledge base at each firing. Given a listing of the rules and a large amount of time, it is possible to identify how the intelligent system came to a specific conclusion. The operator's understanding of overall system reasoning, however, is not improved by this approach.

Figure 4-20. BEFORE: Example Illustrating Difficulty in Achieving Shared Understanding of Situation between Operator and Intelligent System

Problem: Promoting Shared Understanding of Situation

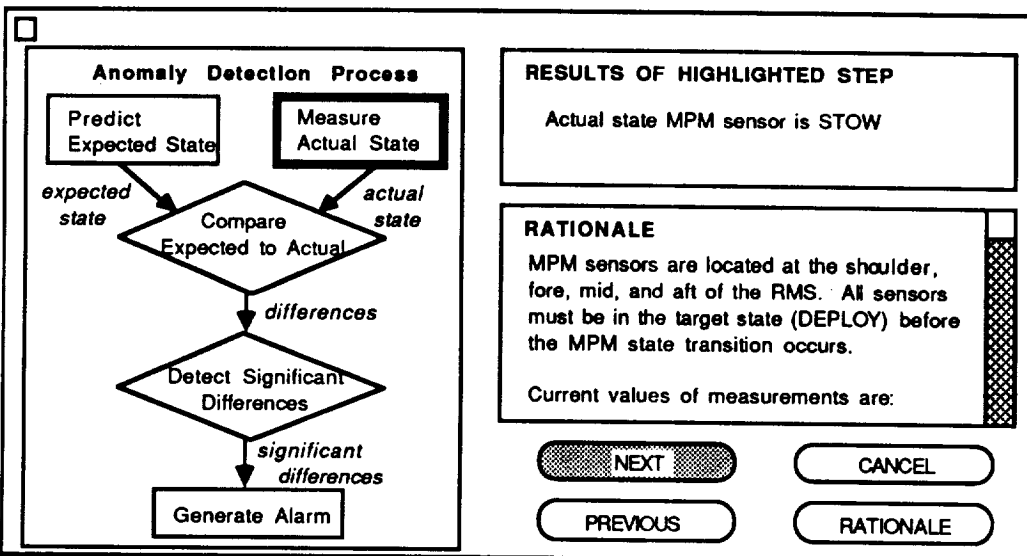
AFTER - Example Illustrating Solution:

The reasoning process of the intelligent system is explicitly represented to assist in establishing a shared understanding of the current situation as well as familiarizing the operator with this process.



STEP 1

The reasoning process of the intelligent system is represented in the diagram to the left. The operator can step through this process using the control buttons. The first step of the Anomaly Detection Process is illustrated in this figure, including the rationale for the expected state



STEP 2

In this illustration, the operator has used the NEXT button to move to the second step of the Anomaly Detection Process. This illustration continues until the operator understands why the intelligent system generated the alarm.

Figure 4-21. AFTER: Example Illustrating Shared Understanding of Situation between Operator and Intelligent System

Problem: Improving Explanation for Agent Collaboration

Recommendation: Retrospective dialog is insufficient for explanation during real-time monitoring and fault management. The intelligent system must assist the operator in formulating a response to anomalous situations as these situations develop.

Chandrasekaran has identified three types of explanation (Chandrasekaran et al., 1989). Most explanation facilities fall into one of two of these categories: explaining how data match knowledge (e.g., in its simplest form, a rule trace) or explaining knowledge (e.g., in its simplest form, rationale in the form of pre-defined text blocks). The third type is explaining the problem-solving strategy of the intelligent system itself. Much research is needed for all three types of explanation. For example, while pre-defined blocks of text can often be woven together to generate simple situation-specific explanations, a generative capacity is needed to respond when the user does not comprehend the initial explanation. Making intelligent systems into real team players will require them to include some capabilities to model their colleagues' knowledge and problem-solving skills.

Often, explanation systems assume that the operator is wrong and that the intelligent system, as the most knowledgeable agent, has the task of correcting human understanding (Wick, 1989). Human-computer collaboration, however, requires mutual exchange of information to achieve a common understanding of situation between agents. Collaboration is interactive and dynamic, where the perceptions of both participants may change in the course of the discussion. Research into explanation that promotes collaboration is required.

Issue: Explanation formulated to correct or train the user is not sufficient for agent collaboration. Research is needed into techniques that promote exchange of information and enable a shared viewpoint between agents.

Research relevant to agent collaboration issues spans the fields of HCI and artificial intelligence. Within the HCI field, Woods, Roth, and Bennett (1990) have proposed a joint human-computer cognitive system, with collaboration that parallels two human, partial experts working together to gain mutual understanding about a problem (see also section 3.2.2 and appendix C). Also from the HCI perspective, Johns (1990) has investigated the use of schematics (graphic forms) to support explanation for agent collaboration. Based on his experience in developing intelligent systems, Wexelblat (1989) has proposed a broader concept of explanation and defined types of dialog that would assist in collaboration (see table 4-3).

Table 4-3. Queries Promoting Agent Collaboration (Wexelblat, 1989)

<p>Queries Promoting Agent Collaboration</p> <ul style="list-style-type: none">• How do I do what you ask me to do? (i.e., provide instructions)• Why do you ask me to do this task? (i.e., define purpose of task)• How did you come to this question or conclusion? (i.e., clarify your reasoning process or strategy)• By what steps did we get here? (i.e., delineate the history of recent operations)• What do I do next? (i.e., define tasks allocated to the human and intelligent system)• What do you know about? Do you know about X? Can I do Y? (i.e., machine recognizing its limits, what it doesn't know)

Fischer (1989) has evaluated the dialog between humans with different experience levels (i.e., a novice and an expert). This evaluation provides interesting insight into mechanisms for dialogue between human and computer agents. For example, the less experienced agent may require assistance in articulating questions or may require more information than is provided in the original answer by the experienced agent. When expertise levels differ, misconceptions are possible. One of the agents may perceive that the other has misunderstood him and take some action to correct the misunderstanding. Sometimes the more experienced agent will provide information that he perceives to be useful, even if that information has not been explicitly requested.

There are areas in current explanation research that complement the proposed research into human-computer collaboration, including:

- Providing responses to follow-up questions based on the premise that human-human explanation is interactive (Wick, 1989)
- Employing levels of user expertise (i.e., novice to expert) to tailor the content and presentation of explanatory information (Wick, 1989)
- Explaining the problem-solving strategy of the intelligent system using an explicit representation of control strategy in the knowledge base (Chandrasekaran et al, 1989)

Other areas of research that are needed but are currently immature include (Wexelblat, 1989):

- Unsolicited assistance (i.e., an agent does not recognize the need for assistance)
- Machine recognition of the scope and limitations of its knowledge (e.g., What do you know about?)

Most intelligent systems surveyed in the case study has some type of rule trace. Rationale in the form of pre-defined text blocks was observed in the DATA COMM Expert System, GNC

Review

The ability to review recorded information about past situations can assist in establishing a shared agent viewpoint. This approach can be used to provide additional information about the monitored process (i.e., support of fault management) or to clarify the reasoning of the intelligent system (i.e., coordination of agents). Review mechanisms can be as simple as providing a scrolling capability on a message list or as complex as a complete playback of recorded, real-time information with annotations provided by the intelligent system. For the purposes of human-computer collaboration, a capability closer to this second type of review is needed. Review using playback of recorded information can be very effective in clarifying both the situation with respect to the monitored process and the behavior of the intelligent system.

Problem: Providing Access to Event History

Recommendation: The operator should be able to review the sequence of events leading to the diagnostic conclusions and recommendations made by the intelligent system. This review should include access to information and activity sequences used by the intelligent system during its reasoning process.

Example: The example in figure 4-22 illustrates limitations arising from inadequate ability to review information. Figure 4-23 illustrates an improved review capability which allows the operator to use real-time displays to review information. This example is based on Scenario 2.

There are two approaches to reviewing previous information that support agent collaboration. *Playback* consists of re-displaying the output of a previous execution of the intelligent system. It should be distinguished from *replay*, which is the re-execution of the intelligent system using recorded input information. In *replay*, the output information driving the displays is actually generated at the time of the display. Since the operator can alter parameters affecting the execution, the output of *replay* may not be the same as the output of the original execution, even though the input information is the same. *Replay* is also useful for re-executing the intelligent system after restarting from some checkpoint in the past (section 4.1.3).

Playback may be conducted in a variety of ways. The recorded information can be viewed through the same set of displays as were available in real time (i.e., taking another look). In this technique, the operator controls the review process, including the rate of stepping through the recorded information and the display formats that are viewed. This capability was observed in some of the prototypes in the case study (e.g., GNC intelligent systems). An alternate playback technique is to allow the intelligent system to control the display of information and add supplementary information and annotation to focus operator attention on important events and activities. An example of this capability is shown in Johns (Johns, 1990) when discussing the use of schematics to explain a causal relationship.

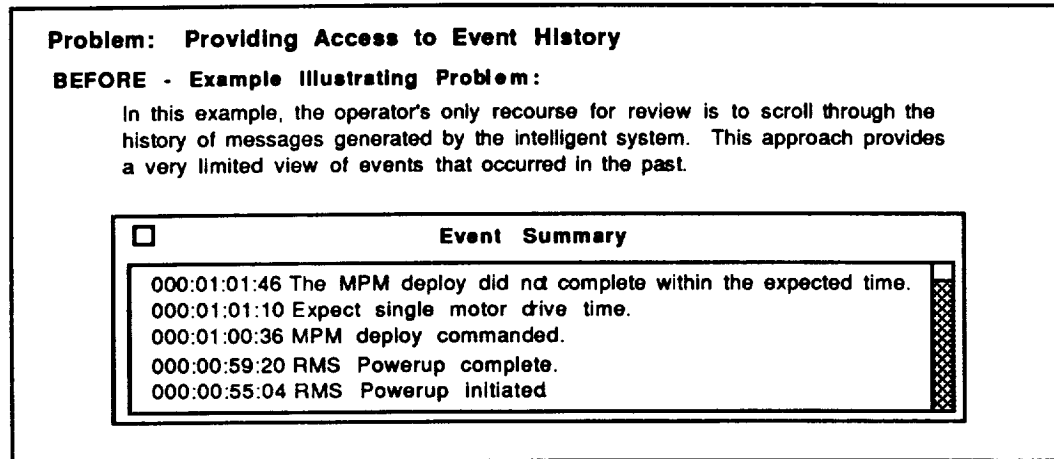


Figure 4-22. BEFORE: Example Illustrating Limited Capability to Review Event History

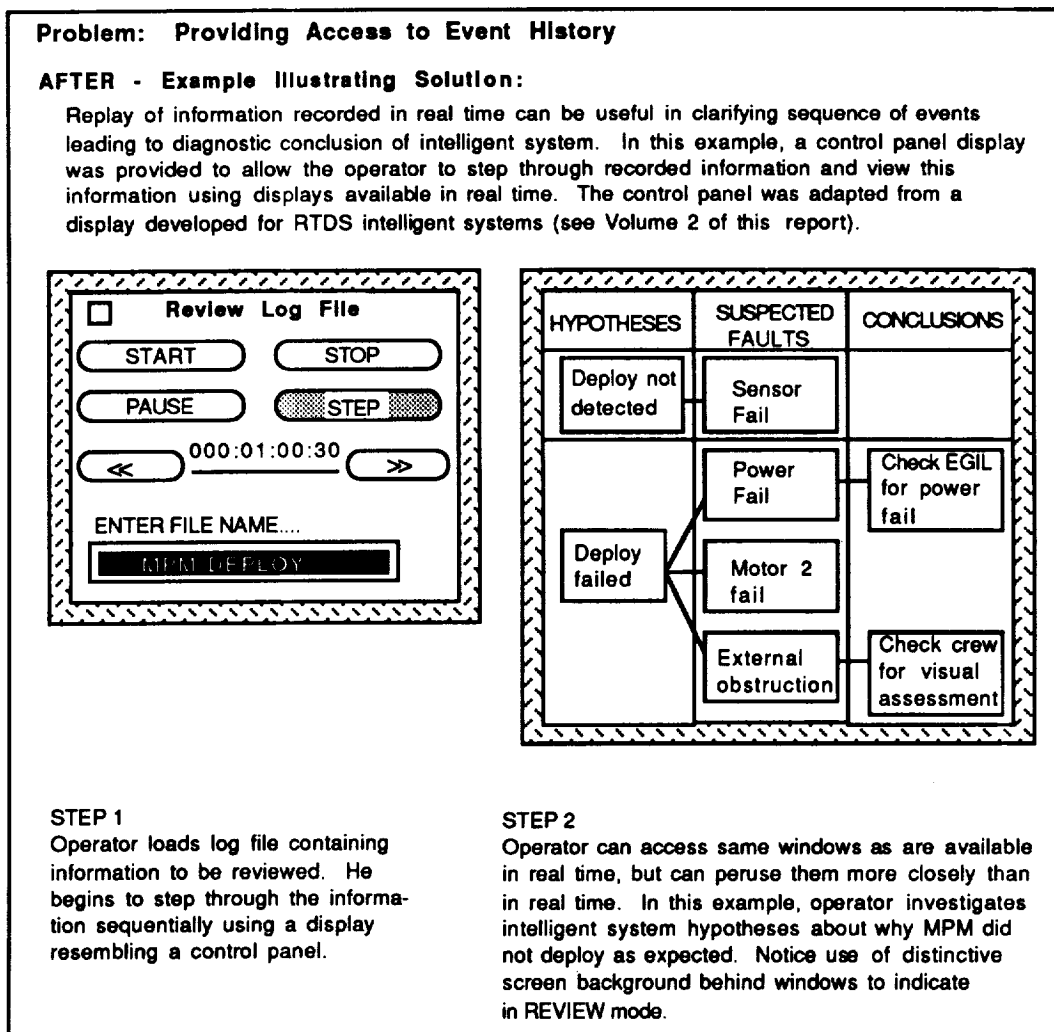


Figure 4-23. AFTER: Example Illustrating Improved Capability to Review Event History

The selection of an approach for information review will affect which parameters are recorded. For playback, information produced by the intelligent system is required. For replay, a log of all information input into the intelligent system is required. Logs of recorded data used for review are called *state sequences* by Johns. An extensive discussion of suggestions for recording and playing back state sequences are provided in Johns (1990). Design issues associated with information logs include (1) identification of the parameters necessary to record while meeting the memory constraints of the system, (2) selection of the sampling rate, (3) identifying the type of control that the operator should exercise over the playback, (4) specification of time periods where information will be recorded, (5) mechanisms for storing information.

Problem: Recording Information for Review

Recommendation: To adequately support review, the designer must define what will be reviewed and how the review will be conducted. This includes determining which parameters should be logged for review. If more than one type of review is provided, the user interface should clearly distinguish the differences between these types and should visually differentiate the displays used for each type.

See figure 4-23 for an example that clearly distinguishes the type of review currently active using a distinctive screen background. See also section 4.1.1 for a discussion of modes.

Regardless of the approach, the review of recorded information should not preclude real-time functionality. Fault management by the intelligent system should continue in the background while the playback is reviewed in the foreground. Additionally, a means of displaying current anomalous behavior observed by the intelligent system that distinguishes between the recorded and real-time information and that attracts the operator's attention should be provided on any displays viewed during the playback.

Review is most effective in explaining behavior that develops over time (i.e., sequence of events). Alternative ways of presenting relevant information may be more appropriate for instantaneous events (i.e., behavior that develops faster than the sampling rate).

Most systems surveyed in the case study had some form of review, with the most common form being a message list. See Volume 2 (Malin et al., 1991) for observations about the use of review in the case study applications.

4.1.3 Managing Intelligent System Errors

The introduction of an intelligent system into flight operations support adds both enhanced capability and an additional source of error. Since the operator is responsible for making the final decisions in fault management situations, the operator must be able to manage the intelligent system. Managing errors in the intelligent system involves two basic activities, determining the source of erroneous intelligent system behavior and correcting or compensating for the identified problem.

Intelligent system errors can be due to failures of the intelligent software or due to erroneous or unavailable input information. Failures of the intelligent software are caused by inaccurate or incomplete knowledge, incorrect reasoning, or inadequate performance. These errors can often be identified during the operational support period (i.e., on-line) if the proper information is provided to the operator.

Typically, errors in the intelligent system result in shut down of the intelligent system. Error correction is performed off-line, after the operational support period. The intelligent system can be designed, however, to permit on-line compensation for errors by operator intervention into intelligent system processing. To effectively intervene in intelligent system processing, the operator must first have a good understanding of the knowledge representation and reasoning strategy used by the intelligent system (section 4.1.2). See also the discussion of the risks of intervention into intelligent system processing later in this section

Methods for intervention into the intelligent system reasoning process vary from small changes in information to a complete takeover of task responsibilities from the intelligent system. Forms of redirection that are discussed in this section include introduction of additional information or changes to existing information, redirection of the reasoning process, restart of the intelligent system, and selective override of portions of the intelligent system processing. For situations when redirection is ineffectual, operator takeover from the intelligent system is discussed.

Detection of Intelligent System Errors

Erroneous intelligent system behavior indicates either a problem in the information being provided to the system or a problem in the intelligent system software (excluding failures of hardware platform and system services (e.g., operating system)). The detection of software errors requires that adequate information about the intelligent system be provided for the operator to identify the source of the anomaly. Erroneous intelligent system behavior can include:

- Drawing an incorrect conclusion due to:
 - Bad input data (errors in data source or errors in transmission)
 - Errors or omissions in the knowledge base
 - Errors in the reasoning mechanism (i.e., the underlying tool/shell or language used to build the system)
- Being unable to draw a conclusion due to:
 - Investigating an unproductive hypothesis
 - Failing to investigate a likely hypothesis
 - Inability to process data within a pre-specified cycle time (in real time)

The recommended approach for detecting software errors is to provide ready access to the internal information and reasoning of the system (section 4.1.2).

Problem: Identifying Intelligent System Errors

Recommendation: Identify and provide access to the critical information from the intelligent system that assists in identifying the source of erroneous software behavior. This includes information about intelligent system knowledge representation, reasoning process, and performance.

For complex systems, a large amount of information about the intelligent system may be required. Providing access to all of this information all of the time may become impractical and actually complicate operational support by introducing an information overload problem. See sections 4.3 and 5 for a discussion about designing to manage information overload.

Intervention by Altering Information

One approach to intelligent system error compensation is altering the information used by the system. This can be information about the monitored process or the intelligent system. Information from the monitored process may be altered to accommodate noisy or unavailable data. The ability to intervene in intelligent system processing by altering the system's internal information can help mitigate system failures due to an incomplete knowledge base. Alteration can include changes to existing information or introduction of additional information (e.g., provide information that is unavailable in electronic form). See the discussion of the risks of intervening in intelligent system processing later in this section.

Problem: Altering Information to be Used by Intelligent System

Recommendation: The operator should be able to alter the information that is processed by the intelligent system. This includes information from the monitored process (Johns, 1990) as well as hypotheses and conclusions of the intelligent system. Alteration should include the ability to input additional information and delete or modify incorrect information.

The PDRS HCI design concepts included the capability to alter the information processed by the intelligent system (figure 4-24). See Volume 2 (Malin et al., 1991) for a description of this capability.

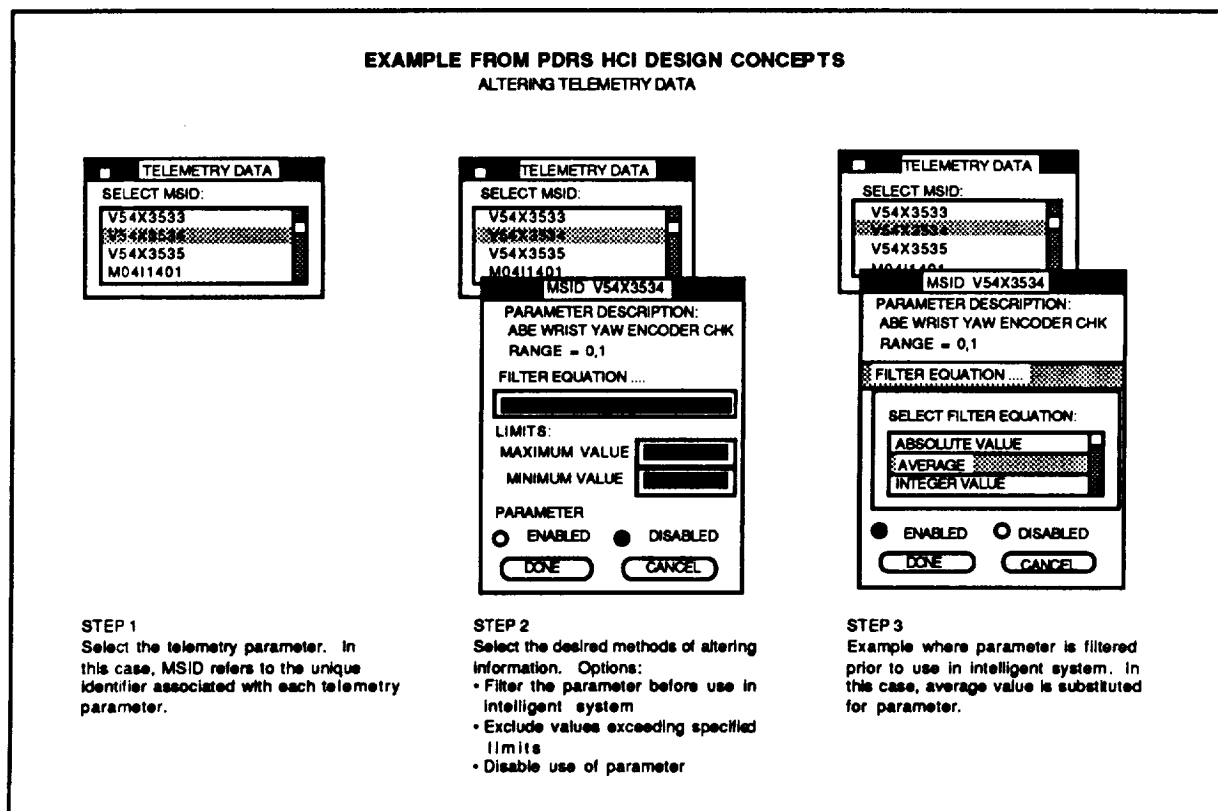


Figure 4-24. Example from PDRS HCI Design Concepts Illustrating Alteration of Information Used by Intelligent System (Schreckenghost, 1990)

Intervention into Reasoning Process

In addition to altering the information processed by the intelligent system, the operator can alter the way that the information is processed. Such an approach would be used to redirect a "disoriented" intelligent system (e.g., the system investigating an unproductive hypothesis or failing to investigate a likely hypothesis) onto a more productive path of investigation. Methods of intervening in the method of processing information include modification of the reasoning process or selection of an alternate reasoning mechanism. Examples of modification of the reasoning process are (1) setting processing priorities (e.g., what hypothesis to investigate first, priorities for knowledge base search), (2) alteration (modify, add, or delete) of hypotheses, and (3) specification of alternate solutions.

Problem: Redirecting Intelligent System Reasoning

Recommendation: Operator intervention into the reasoning process of the intelligent system can be used to manage errors in the intelligent system. Methods of intervention include modification of the reasoning process or selection of an alternate reasoning mechanism.

Example: The example in figure 4-25 illustrates the difficulties arising when the intelligent system is pursuing an unproductive path of reasoning. In this example, the operator must perform a restart to alter the reasoning. In figure 4-26, the operator is able to alter the intelligent system fault hypotheses, resulting in a return to normal processing. This example is based on Scenario 2.

The user interface used to intervene should reinforce the operator's understanding of the reasoning process to assist in identifying appropriate action. See section 4.1.2 for a discussion of presenting information about intelligent system reasoning. See the discussion of the risk of intervening in intelligent system processing later in this section.

Restart of the Intelligent System

The use of restart to redirect an intelligent system is a destructive method. All information processed and derived up to the time of the restart is lost. Restart should only be used in situations where the current processing strategy cannot be redirected by less drastic means. Such situations include significant errors in or loss of input information or errors in the intelligent system processing strategy.

It is possible to design the intelligent system to minimize information loss at restart. Such a capability requires periodic storage of intelligent system state information (i.e., *checkpoint*) and information input to the intelligent system. A checkpoint is the storage of all intelligent system internal information at a given time. The checkpoint data can be reloaded into the intelligent system and system processing can be resumed from the checkpoint. The intelligent system is re-executed (i.e., replayed) using the recorded input information. If the intelligent system is part of a distributed system, using checkpoints is more complex. Other mechanisms besides time (e.g., events) may be more effective for synchronizing restart of distributed systems.

Checkpointing can be a useful capability for redirecting the intelligent system, because it permits reprocessing of data after needed alterations have been made by the operator. Such a capability is useful when erroneous input has been processed or the intelligent system knowledge base is flawed.

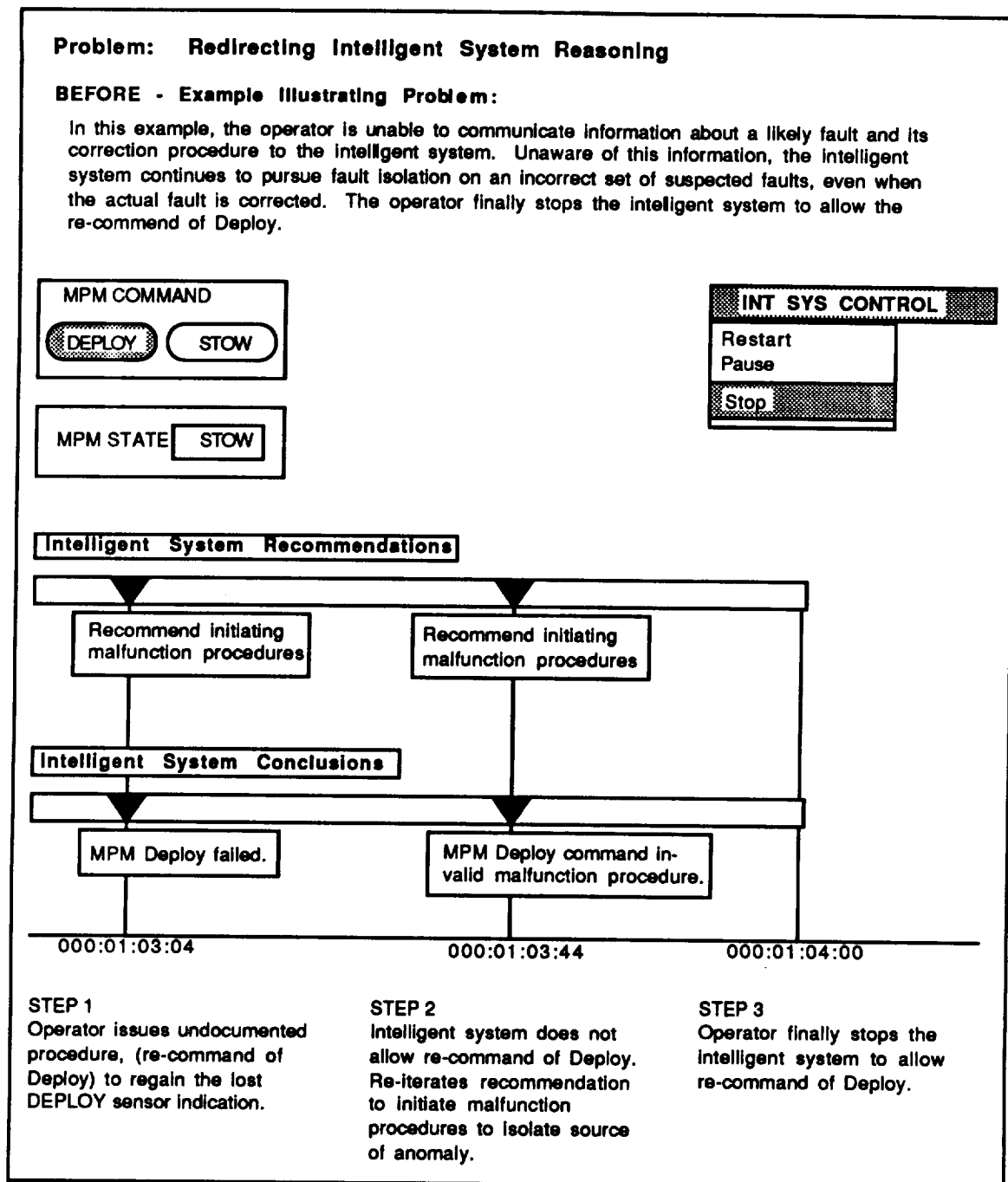


Figure 4-25. BEFORE: Example Illustrating Difficulty When the Intelligent System Becomes Disoriented

Problem: Redirecting Intelligent System Reasoning

AFTER - Example Illustrating Solution:

In this example, the ability to affect intelligent system reasoning by altering hypotheses used by the intelligent system has been provided to the operator. The operator "informs" the intelligent system about the identity of the actual fault (sensor failure) and a corrective procedure to "fix" the fault (re-command of Deploy). The redirected intelligent system can now continue normal processing.

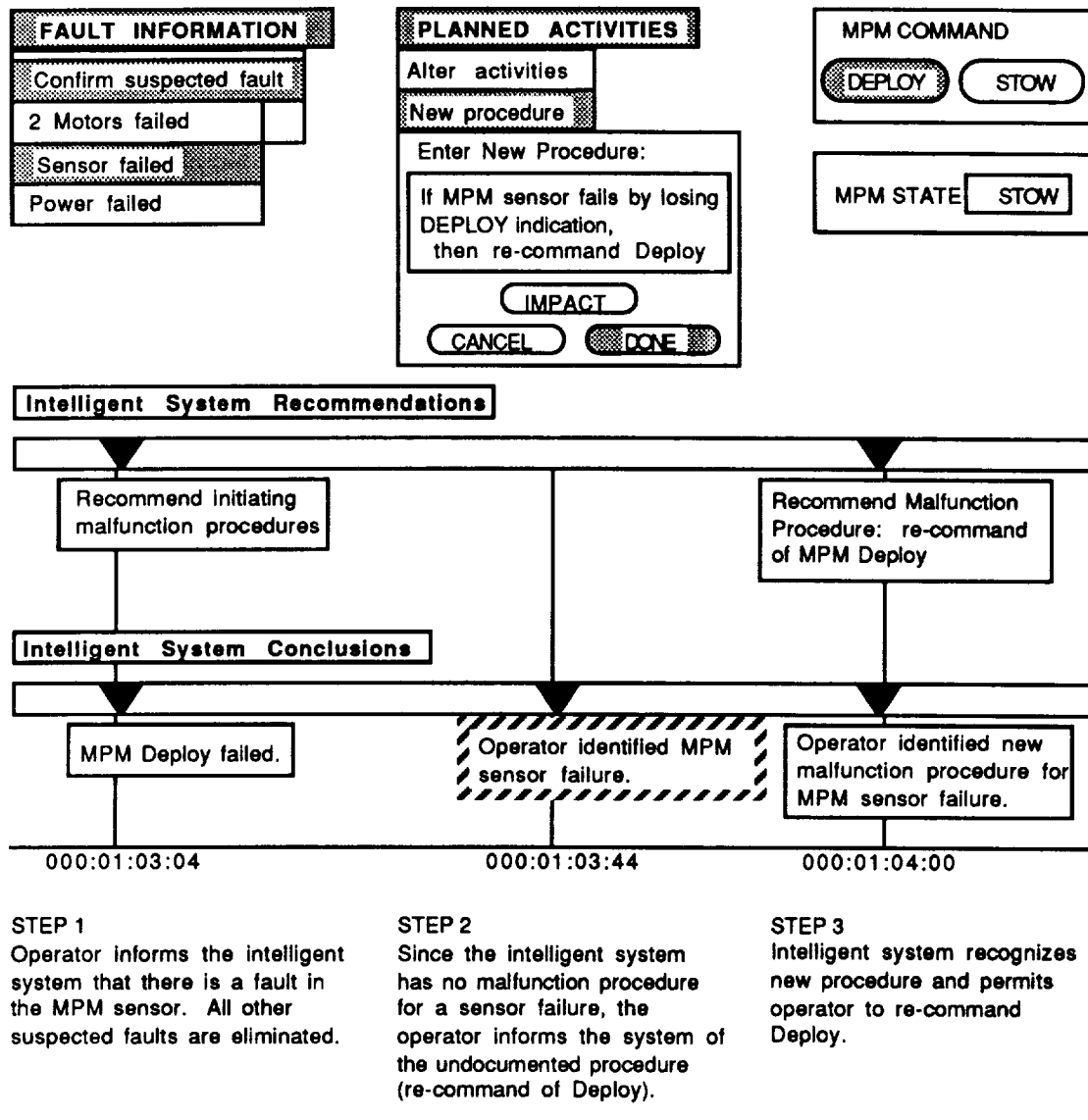


Figure 4-26. AFTER: Example Illustrating Operator Redirecting Disoriented Intelligent System

Problem: Support for Intelligent System Restart

Recommendation: Intervention capability should include restarting the intelligent system from a saved state (i.e., checkpoint) and replay by executing the intelligent system using recorded input data. This allows the operator to alter the information used by the intelligent system (both internal and external) and re-execute from some time in the past.

Part of providing checkpoint capability is determining when a checkpoint will be taken. Checkpoints can be manual or autonomous. Autonomous checkpoints can be taken periodically or when pre-defined events occur. For example, it would be useful to take a checkpoint prior to major perturbations in either the monitored process (e.g, change software load onboard the Space Shuttle) or the intelligent system (e.g., unavailable telemetry due to Loss of Signal). Figure 4-27 illustrates the different types of checkpoint capability provided in the PDRS HCI design concepts.

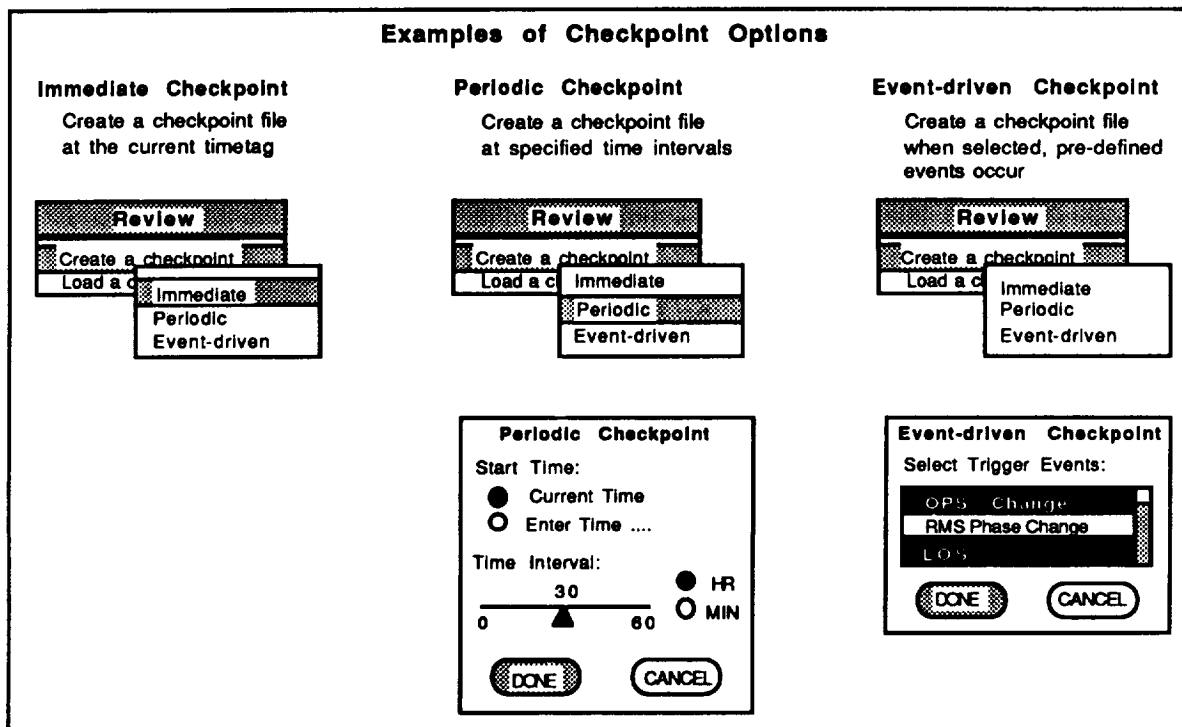


Figure 4-27. Checkpoint Options Provided by PDRS HCI Design Concepts (Schreckenghost, 1990)

Problem: Recording Input Data for Replay

Recommendation: The ability to replay the intelligent system using recorded data requires that the intelligent system have a source of recorded input data. If an external source of such information is not available in real time, the intelligent system should provide the ability to archive such data. This ability includes control of the data archiving from the user interface.

Notice that the ability to replay the intelligent system using recorded data has some off-line benefits as well. Such a capability can be used for verification and validation by replaying the system using test cases of recorded input data. Replay can also be used as a source of stand-alone training for new operators.

The ability to restart the intelligent system was the most common, and often the only, method of intervening in intelligent system processing observed in the case study.

See section 4.1.2 for related recommendations on the use of replay and playback for review of information supporting agent collaboration. See also the discussion of the risks of intervening in intelligent system processing later in this section.

Override of the Intelligent System

An alternate method of redirection is selective override of segments of intelligent system processing. Selective override is the process of altering the responsibilities of the intelligent system by providing variable operator control over intelligent system tasks. Override can be specified with a wide range of authority for the operator, from limited operator involvement in intelligent system responsibilities (i.e., some level of operator confirmation) to full control by the operator with no intelligent system assistance (Johns, 1990).

Various approaches to selective override can be used. Modes of operation (section 4.1.1) can be defined that specify multiple levels of responsibility for the intelligent system (e.g., see table 4-4 for an example of procedure execution modes from the OMA prototypes). Alternately, portions of the knowledge base can be enabled or disabled to allow the operator to customize intelligent system knowledge to the current situation (e.g., partition knowledge bases by mission phase). Knowledge base partitioning can be used to improve intelligent system performance by off-loading unnecessary portions of the knowledge base or to manage intelligent system failures by disabling the erroneous portions of the knowledge base.

Table 4-4. Example of Procedure Execution Modes from the OMA Prototypes (Kelly, 1991)

Five Levels of Automation for Procedures Execution

- **Hardware Switch Manual**
Execution of procedure by operator
- **Software Switch Manual**
Execution of procedure by intelligent system at operator request
- **Automatic with Confirmation**
Step-wise execution of procedure with operator confirmation at each step; if hardware switch, operator will execute and if software switch, intelligent system will execute
- **Full Auto:** execution of procedure by intelligent system with no operator intervention required
- **"Start-Stop":** execution of procedure as designated, where each execution step is pre-specified as one of the above types

Problem: Support for Selective Override of Intelligent System Processing

Recommendation: Selective override of specific intelligent system activities can be used to intervene into intelligent system processing to improve system performance or to control the effects of errors in the knowledge base. Techniques for selective override include disabling portions of the knowledge base or altering modes of operation to control allowable activities.

Example: In the example in figure 4-28¹, the knowledge base of the intelligent system does not contain the required knowledge about a malfunction procedure. This results in the intelligent system pursuing an unproductive path of reasoning. In figure 4-29, selective override of a portion of the knowledge base provides a less drastic recovery approach than performing a restart. This example is based on Scenario 2.

The ability to perform a selective override affects the design of the intelligent system. Modes of operation that specify allowable intelligent system activities must be explicitly provided. If selective enabling and disabling of the knowledge base is required, the knowledge base must be designed such that portions can be loaded or unloaded during execution. Although selective override would typically be performed by the operator, it is possible for a higher level (metalevel) knowledge base to coordinate the selection of mode or the enabling and disabling of subsystem knowledge bases.

¹ Notice that figure 4-28 is an exact duplicate of figure 4-25. This set of examples represents an alternate approach to the problem posed when discussing intervention into intelligent system reasoning. Figure 4-25 was duplicated to allow easier comparison between the BEFORE and AFTER cases.

Problem: Support for Selective Override of Intelligent System Processing

BEFORE - Example Illustrating Problem:

In this example, a situation previously considered is revisited and an alternate solution is described. Previously, in figure 4-25, we illustrated the situation where the operator is unable to communicate information about a likely fault and its correction procedure to the intelligent system. Unaware of this information, the intelligent system continues to pursue fault isolation on an incorrect set of suspected faults, even when the actual fault is corrected. The operator finally stops the intelligent system to allow a re-command of Deploy. Note that the BEFORE case is a repeat of figure 4-25.

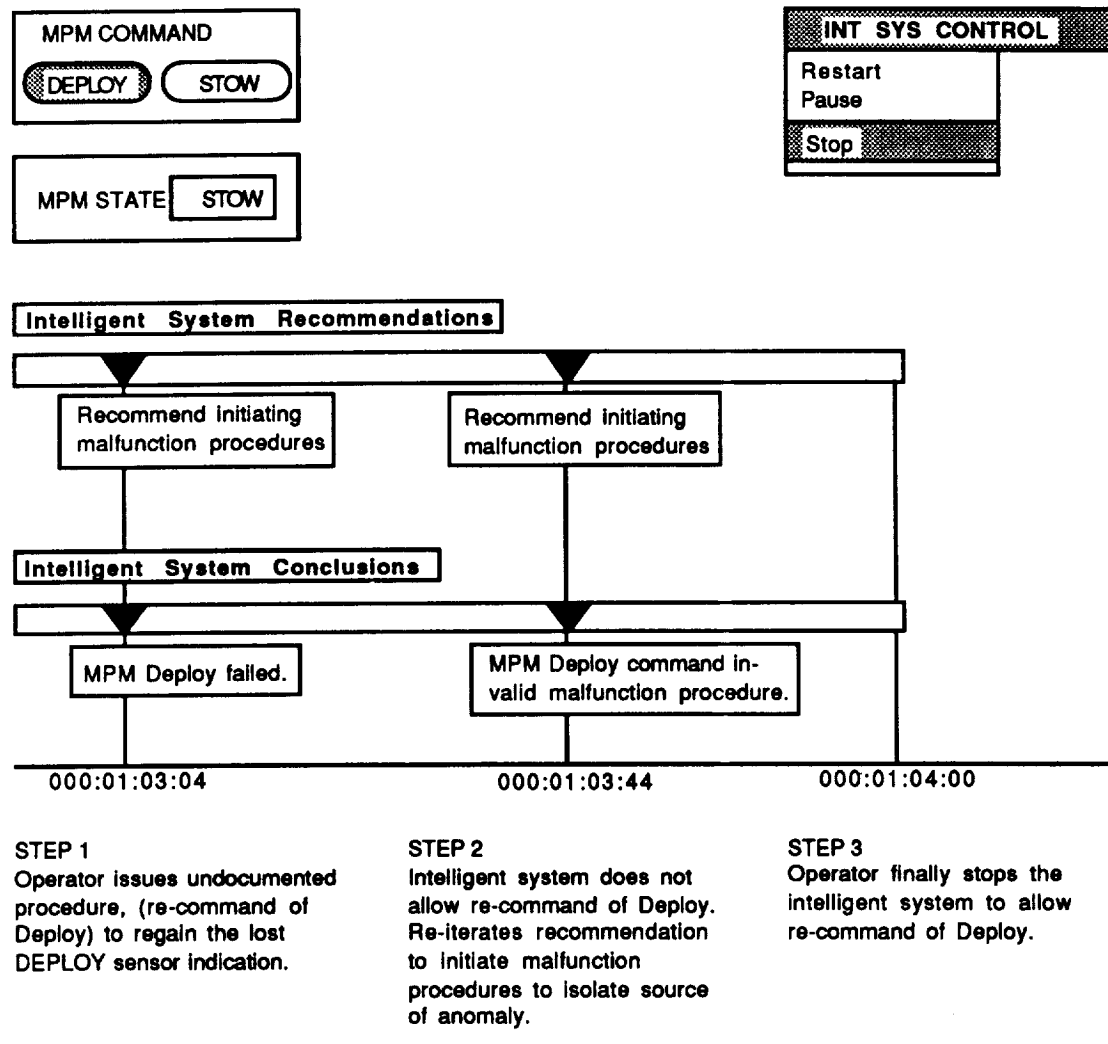


Figure 4-28. BEFORE: Example Illustrating Intelligent System with an Omission in its Knowledge Base

Problem: Support for Selective Override of Intelligent System Processing

AFTER - Example Illustrating Solution:

In this example, the ability to selectively disable rule sets used by the intelligent system has been provided to the operator. The operator disables the MPM rule set and proceeds with the MPM Deploy. The intelligent system will resume normal processing when the next phase of operations initiates (MRL release).

MPM COMMAND

DEPLOY
STOW

ENABLE RULE SETS

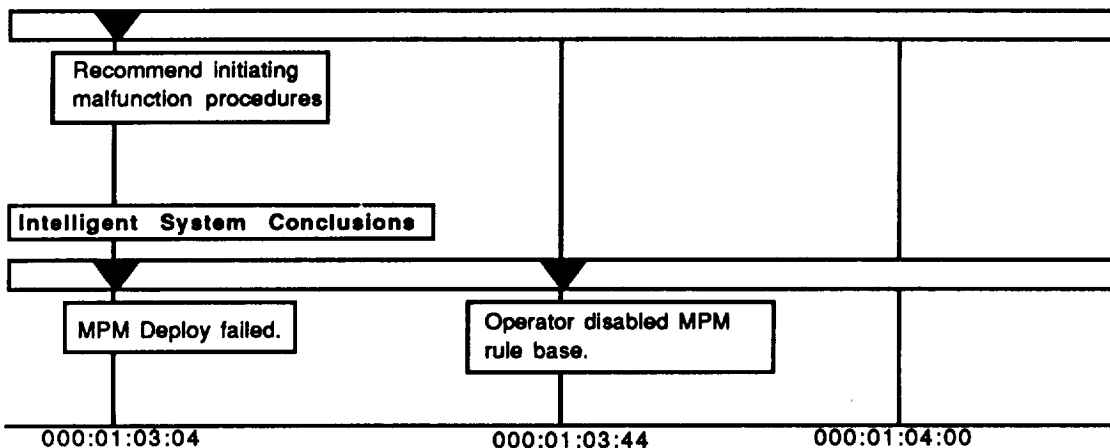
MPM	DISABLE
MRL	ENABLE
MCIU	ENABLE
ABE	ENABLE
EE	ENABLE
D&C	ENABLE

DONE
CANCEL

MPM STATE

STOW

Intelligent System Recommendations



STEP 1

Operator issues undocumented procedure to regain the lost DEPLOY sensor indication.

STEP 2

Operator realizes that the MPM rule set is inaccurate. He selectively disables that rule set from consideration.

STEP 3

Once the inaccurate rule set has been disabled from processing, the intelligent system will be on hold until the next phase of operations (MRL release), when normal processing will resume.

Figure 4-29. AFTER: Example Illustrating the Use of Selective Override to Correct for Knowledge Base Error

The PDRS HCI design concepts included the capability to intervene into the intelligent system processing by selectively disabling rule sets. See Volume 2 (Malin et al., 1991) for a description of this capability.

In situations where the intelligent system fails unrecoverably, fault management tasks must be accomplished without any assistance from the intelligent system. In such an event, the operator should be able to takeover all intelligent system responsibilities. This corresponds to the level of override "full control by the operator" discussed above (see Hardware Manual Switch mode in table 4-4).

Implicit in the decision to allow the operator to disable the intelligent system is the requirement that critical operational information can be displayed independently from the intelligent system. This can be accomplished by either having a display-only mode of operation for the intelligent system or by having separate displays that provide access to information and control of the monitored process. Operators should be trained to take over from the intelligent system to prevent over reliance on the intelligent system which can result in operators incapable of performing the task alone. The design of the entire support system (including both conventional and intelligent software) should support fault management without the intelligent system. One approach to achieving such a design is to start off with no intelligent system in the design and only add the intelligent system after designers have addressed fault management without it. See also section 4.2.2 on visibility into the monitored process.

Problem: Support for Complete Override of Intelligent System

Recommendation: The operator should be able to completely take over control of all activities normally performed by the intelligent system. To allow a complete override of the intelligent system by the operator, critical telemetry and information derived for problem diagnosis and recovery of the monitored process should be available for display independent of the operation of the intelligent system. One approach to achieving such a design is to start off with no intelligent system in the design and only add the intelligent system after designers have addressed fault management without it.

When a complete override of the intelligent system occurs, it may be useful to continue passive background execution of the system. The results of that execution could be logged for use in correcting the software failure and maintaining the intelligent system.

See the example in Section 4.1.1 (figure 4-5) for an illustration of the operator overriding the intelligent system. See also the following discussion of the risks of intervention in the intelligent system.

Risks of Intervention in Intelligent System Processing

There are risks inherent in intervening in intelligent system processing. The effect of an alteration may not be obvious and unintentional impacts can result in some cases. If the intelligent system actually controls the monitored process, it may be difficult to reverse intelligent system actions impacting the monitored process. A cautious attitude should be taken toward the use of such intervention. Intervention should only be undertaken when the operator thoroughly understands the situation and is aware of the potential impacts of such intervention. This awareness could include the intelligent system assessing and informing the operator that it is not a good idea to execute the desired changes at this time. If possible, analytical tools to evaluate the impact of intervention should be provided (see section 4.2.2 for a discussion of

evaluating consequences of events). A checkpoint should be taken prior to intervention to permit reversion to a former state should intervention be ineffective (see discussion of checkpoint in this section).

Problem: Avoiding Negative Impacts from Operator Intervention

Recommendation: If the operator can intervene into intelligent system processing, a means of predicting the impact of this intervention should be provided. Additionally, a checkpoint should be taken prior to intervention to permit reversion to a former state should intervention be ineffective.

4.2 Support for Fault Management

Fault management activities are usually initiated by alarm annunciation. As symptoms of failures, alarms represent important fault management information. Alarms must be associated with a system failure (or set of suspected failures). The interpretation of alarms involves assessing the accuracy with which the alarm reflects the current situation (i.e., identify false alarms) and the severity of the impact of the anomalous behavior on safety or mission goals (i.e., criticality of the associated failure). Section 4.2.1 discusses the management of alarms as part of managing faults in the monitored process.

When fault management is performed by a team of agents, fault management activities are distributed between these agents. For the operator to retain the capability to manage the monitored process and make decisions in failure situations, he must maintain an awareness of the overall state and behavior of the monitored process, even when significant portions of the fault management tasks are conducted by the intelligent system. Section 4.2.2 describes the diagnostic information critical to understanding situations affecting the monitored process. Specific topics include assessment of mission impacts, evaluation of consequences and alternatives, monitoring and execution of procedures, and assessment of functional capability remaining after a failure.

Not all fault management situations can be anticipated. The intelligent system must be designed to assist the operator during contingency or unanticipated situations by providing information required for generating and testing workaround procedures. Section 4.2.3 provides recommendations for designing systems to handle contingency situations.

4.2.1 Alarm Management

Operator workload often increases dramatically during an anomalous situation (appendix B). The anomaly can alter conditions in multiple systems, resulting in a profusion of alarms that must be responded to in a timely manner. The amount of information that must be processed increases and the performance requirements become more rigorous. An important aspect of managing this increased workload is managing the alarms that result from the anomaly.

The management of alarms involves a variety of tasks. Important diagnostic information must be extracted from the wealth of alarm information associated with an anomaly. False alarms must be distinguished from actual anomalies. Redundant, possibly inconsistent alarms must be interpreted. Recommendations for alarm management that address these issues are discussed in this section.

Interpretation of Alarms

An alarm is a notification of an anomalous event. When parameters defining system behavior are outside of an envelope of nominal behavior, an alarm is annunciated. The behavior envelope is usually specified as a set of parameter limits (upper, lower, or both) that bound nominal behavior. There may be multiple sets of limits for a given parameter, depending upon configuration or operating level [performance] of the related system. It may also be necessary to distinguish between parameters out-of-limits and off-scale (JSC, June 1990). For out-of-limit parameters, the parameter value is a true reading of current state. For off-scale parameters, the limits of the measuring device are exceeded and the parameter value is set to the limit of the sensing device. See table 4-5 for a definition of alarms used for the Space Shuttle Caution and Warning (C&W) System.

Table 4-5. Alarm Definition for Space Shuttle Caution and Warning (JSC, June 1990)

Alarm Definition for Space Shuttle Caution and Warning	
• Limits	The set of parameter values that define the boundary between nominal and anomalous behavior. Each set consists of a lower and upper limit. Multiple sets of limits are possible for some types of alarms, with only one set active at any time.
• Noise Filter Values	The number of consecutive samples out-of-limit for alarm annunciation and alarm disable.
• Annunciation Enable/Inhibit Status	The parameter status indicating if a detected alarm condition should be annunciated.

Before responding to an alarm, the fault management team must ascertain the accuracy of the alarm. In some cases, it may be possible to eliminate the alarm as false. In other cases, an indicator of alarm reliability and accuracy may be provided to assist in interpreting the alarm. Methods for assessing the accuracy and reliability of an alarm include:

- **Ascertain if other indicators of system behavior are consistent with alarm**
If other parameters are available for monitoring system behavior than the parameters used to activate the alarm, they can be monitored for behavior consistent with the alarm state. For example, low oil level in an engine can be indicated by both a low oil pressure and increased operating temperature.
- **Associate a confidence measure with an alarm**
For example, the percentage agreement among redundant alarms can be a measure of confidence.

- **Evaluate previous behavior for trends**
Trends can include an observed history of frequent system failures of this sort or a history of false alarms.
- **Delay response to determine if behavior is transient**
Duration of delay depends on criticality of failure and permanence of the alarm condition
- **Verify related system is in the expected configuration**
System misconfiguration can appear to be a failure by resulting in unexpected behavior.

False alarms are discussed later in this section. See this discussion for assistance in managing false alarms.

Problem: Indicating Alarm Accuracy

Recommendation: Inaccurate alarm information should be eliminated from consideration by the fault management team when possible. If the reliability or accuracy of the alarm is in question, this should be indicated to the fault management team. Information useful in assessing the accuracy of alarms includes agreement between redundant indicators, previous behavioral trends, duration of an alarm, and system configuration.

Alarm situations, particularly emergency situations (e.g., threat to crew or vehicle safety), are often accompanied by a flood of information as the anomalous behavior propagates through the monitored process and related systems. The fault management team must determine which alarms should be responded to first. The criticality of the failure indicated by an alarm should be considered when planning response to the alarm. Alarms may also indicate the potential for a failure to propagate throughout the system and cause other failures. Table 4-6 describes the classes of alarms used for Space Shuttle to indicate the severity and importance of an alarm. Later in this section, in the discussion of false alarms, figure 4-31 provides an example that includes disagreeing, redundant indicators (e.g., discrepancy between sensor 1 and sensor 2). See also section 4.2.2 for a discussion of critical function loss due to system failures, section 4.1.1 (interruption of operator) for methods used to prioritize messages by criticality, and section 3.4 for a discussion of criticality

Table 4-6. Alarm Classes for Space Shuttle Caution and Warning (JSC, June 1990)

Alarm Classes for Space Shuttle Caution and Warning

- **Emergency**

Two types of alarms are in this class: smoke detection/fire suppression and rapid cabin depressurization.

- **Caution and Warning**

Includes both primary Caution and Warning hardware system and backup Caution and Warning software system.

- **Alert**

Notifies an impending Caution and Warning alarm or a situation requiring a long (exceeding 5 minutes) procedure to rectify problem.

- **Limit Sense**

Indicates a parameter out of limits or in an off-nominal state.

Problem: Distinguishing Severity of Alarms

Recommendation: The severity of an alarm should be used to focus operator attention on important diagnostic information and assist in planning anomaly response. Alarm severity can be assessed based on the impact of the anomalous behavior and associated failure, including the potential for the failure to propagate through the system.

If the alarm indicates a possible emergency, it should be acted upon quickly with action to minimize risk, even if there is the possibility of a false alarm. The additional alarm processing required to detect false alarms can introduce time delays. When an alarm indicates an emergency, it may be required to by-pass all alarm pre-processing and immediately annunciate the alarm (e.g., Space Shuttle automatic safing procedures when crew cabin pressure drops significantly (JSC, June 1990). See also section 3.1 for additional discussion of safing.

In addition to the accuracy and severity of an alarm, the following information about an alarm can be useful in interpreting the alarm:

- **Set of possible failures that could cause the anomalous situation**
An alarm may be activated by more than one failure. In such cases, the alarm is associated with a set of possible failures and additional diagnosis is required to identify the fault. See also the discussion of fault ambiguity under quality of information in section 4.3.1.1.

- **Sequence in which alarms occurred**
If multiple alarms are activated, the alarm sequence may convey information useful in identifying the failure causing the alarm condition. This is especially true when failures cascade across multiple components or systems (i.e., failure A causes failure B) (appendix B). See section 4.2.2 for a related discussion of failure propagation.
- **Source of alarm**
Alarm information can be grouped by physical or structural relationships between the data sources activating the alarms. This approach reinforces the operator's model of system structure and clarifies the meaning of alarms. See section 4.3.1.1 for a discussion of source of information.
- **Procedures for failure compensation and recovery**
Once a failure has been identified, these procedures are useful in planning failure response. See also section 4.2.2 concerning the assessment of mission impacts and procedures.

Problem: Relieving Operator Overload due to Multiple Alarms

Recommendation: The large amount of information generated during an alarm situation can overwhelm an operator trying to interpret that information. The human-computer interface should assist the operator in associating alarms with failures and in planning response to alarms. Specific information for failure identification includes the set of suspected failures, the sequence in which the alarms occurred, and the source of alarms. Information useful in planning alarm response includes the confidence in the accuracy of the alarm, the severity of the alarm, the identity of the failure (or set of failures), and the available procedures for failure compensation and recovery.

Some techniques used to manage alarms:

- **Alarm inhibit**
Alarm inhibit can be used to control information overload by disabling irrelevant alarms. It can also be used to eliminate erroneous alarms from consideration by the fault management team.
- **Suppression of repetitive alarms**
A repetitive alarm is one alarm issued multiple times for the same alarm condition. It should be distinguished from a redundant alarm which is a separate, independent alarm that provides redundant information. Suppression of annunciation of repetitive alarms is another means of controlling information overload in alarm situations.
- **Alarm acknowledge**
Alarm acknowledge is a reminding technique for the operator. It distinguishes alarms recognized by the operator from those not yet examined. The designer should be cautious in using this feature. If repetitive alarms are not suppressed, alarm acknowledge can become a severe load on the operator.

Fault message and discrete status lights are the most common ways to present alarm information. Color and aural annunciation are frequently used to code criticality. See sections

4.3 and 5 for a discussion of issues related to presentation of information, including message lists.

False Alarms

In the discussion of alarm interpretation in the previous section, the concept of alarm accuracy was introduced. A false alarm is an alarm that inaccurately indicates the existence of an anomaly. The management of false alarms is discussed in this section.

Noisy or failed sensors are common in aerospace control systems. The erroneous data from such sensors can generate false alarms and misdirect the intelligent support system that processes the erroneous data. Behavior of the system inconsistent with the annunciated alarm may indicate a false alarm due to sensor error. Alternately, such false alarms may be detected by disagreement between redundant alarms.

The operator should have the ability to manage errors in data from the monitored process. Errors in data can be managed by allowing explicit control of the data sources used by the intelligent system (see the discussion of redundant sources in section 4.3.1.1) and by providing the capability to modify data prior to intelligent system processing (e.g., pre-process information in some way). It may also be desirable to allow the operator to save an altered alarm configuration for re-loading at another time (see section 4.1.3 for a discussion of checkpoint).

Problem: Managing False Alarms due to Bad Data

Recommendation: The ability to manage false alarms resulting from noisy or erroneous data should be provided to the fault management team. Such situations can be managed by allowing the operator to exercise control of what data sources are seen by the intelligent system and to operate on data prior to use in the intelligent system.

Example: The example in figure 4-30 illustrates how bad data can cause false alarms that misdirect the intelligent system. In figure 4-31, the operator excludes the bad data and normal processing continues. This example is based on Scenario 4.

To effectively use such a capability to manage false alarms, the operator must maintain awareness of the modifications performed on the data and be able to reverse the effects of performing such modifications. Incoming operators at a handover must also be made aware of modifications made prior to their support period (section 4.1.1). The ability to reverse the effect of a data modification could include the ability to restart the system from checkpoint (section 4.1.3).

False alarms may also be caused by transient behavior in the system. Transient behavior is a temporary, short duration aberration in normal system behavior. Transients often occur near state changes in a system (e.g., steady state biases may become temporarily time-variant at a state change, such as a momentary loss of system power). A common solution to false alarms generated by transient behavior is to wait for a short time period before generating an alarm to confirm the existence of anomalous behavior (e.g., Space Shuttle C&W noise filter value

Problem: Managing False Alarms due to Bad Data

BEFORE - Example Illustrating Problem:

In this example, 2 temperature sensors are available for each joint of wrist. It is expected that all would sense similar temperatures at a given time. An alarm has been issued based on sensor 1 for roll joint indicating high temperature. Since all other measurements are similar, the operator concludes that wrist roll sensor 1 is bad. The intelligent system, however, evaluates that there is a discrepancy between sensors and recommend further evaluation. The operator has no way to disable use of data from the bad sensor by the intelligent system.

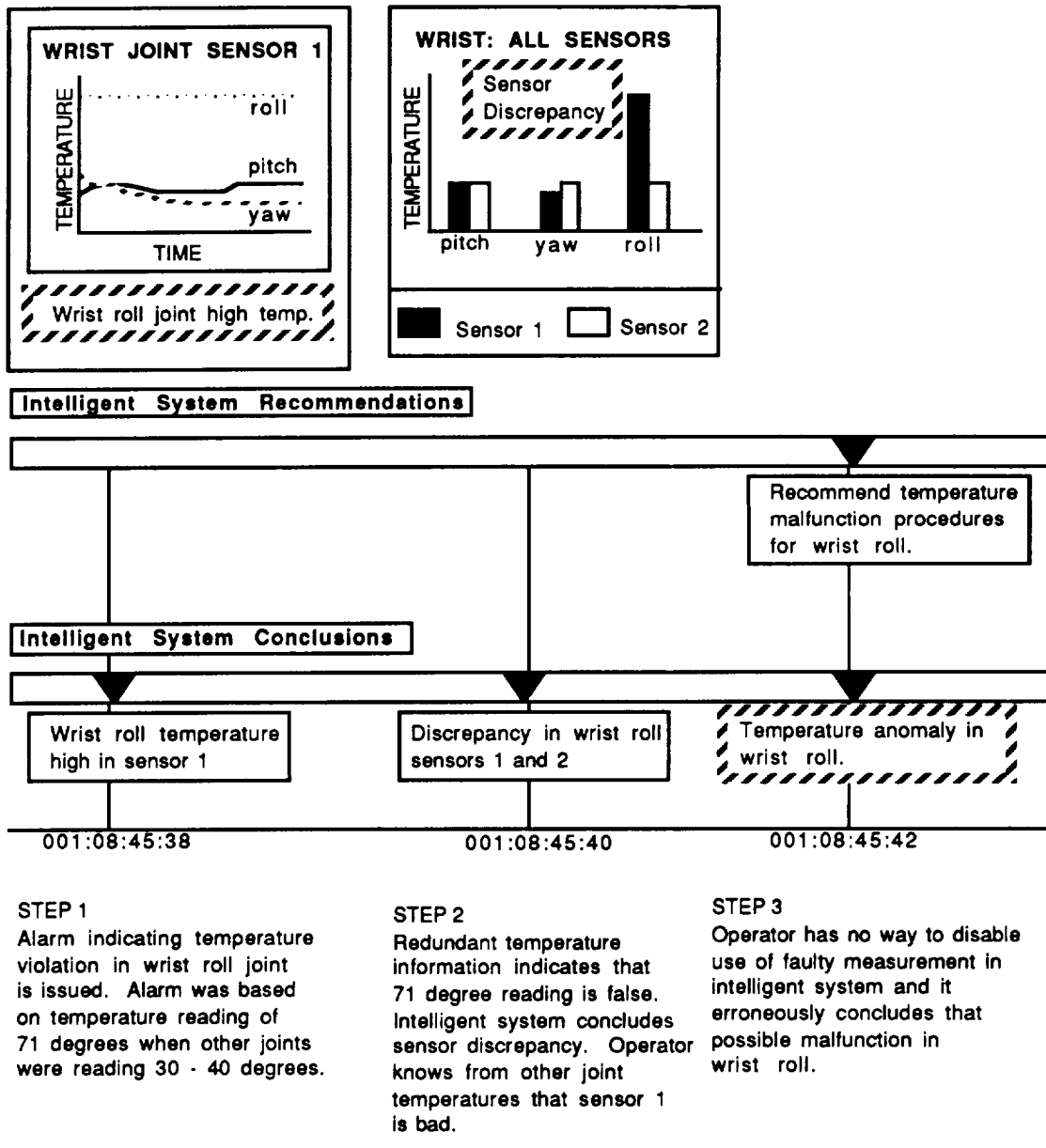


Figure 4-30. BEFORE: Example Illustrating Adverse Affects of False Alarms due to Bad Data

Problem: Managing False Alarms due to Bad Data

AFTER - Example Illustrating Solution:

In this example, the operator disables use of data from the bad sensor by the intelligent system and the sensor discrepancy is resolved.

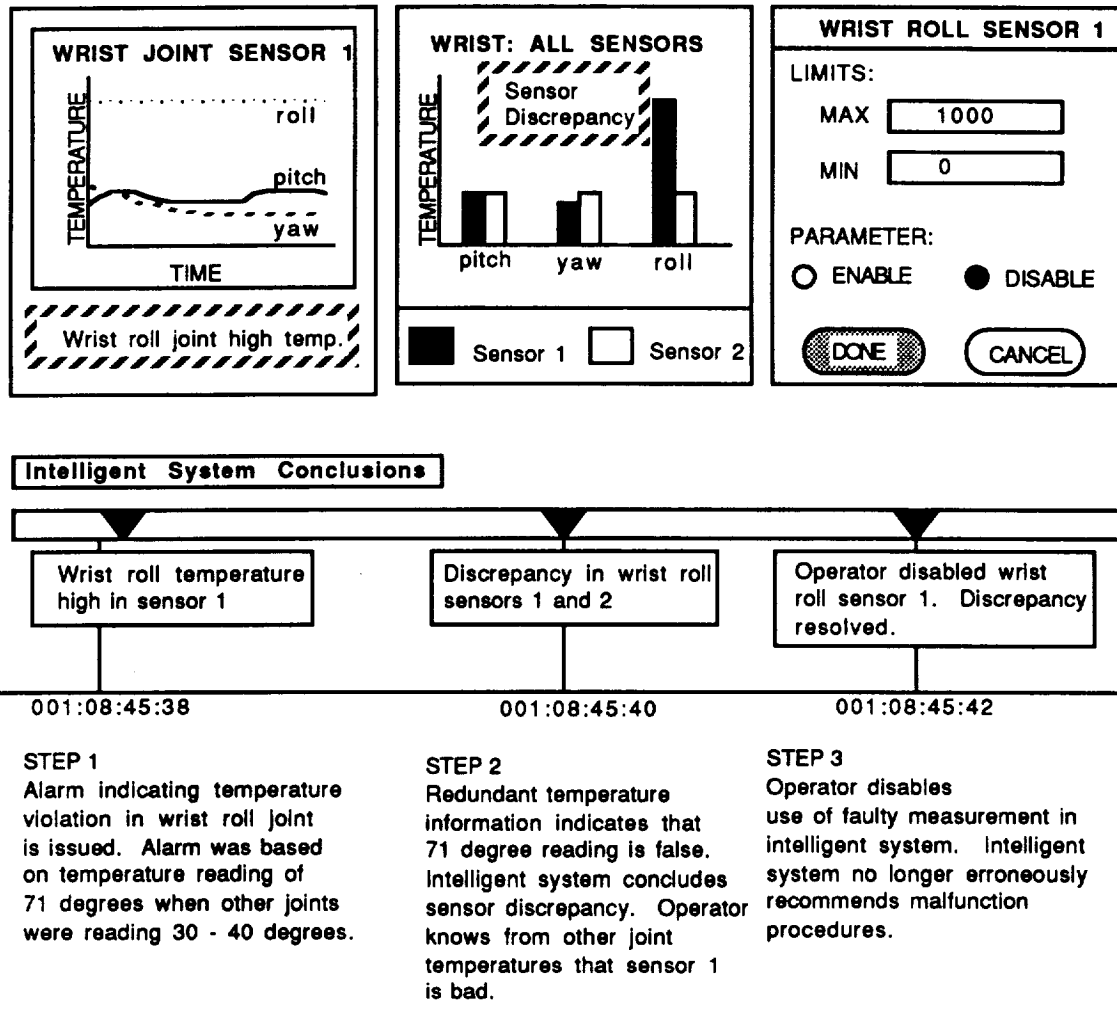


Figure 4-31. AFTER: Example Illustrating Recovery from False Alarm by Excluding Bad Data from Intelligent System

shown in table 4-5). The wait-to-confirm approach is not effective, however, when failure effects can develop and change quickly (i.e., transient behavior is a leading indicator and changes as the failure worsens) or when rapid response to alarms is required (i.e., insufficient time to "wait and see"). An alternate approach is to identify indicators that transience is likely to occur (e.g., recent state change) and rely on alarm information based upon the presence of these indicators.

Intermittent behavior is a type of transient behavior, where normal behavior is periodically interrupted by transient, anomalous behavior. Intermittent behavior often appears inconsistent, due to changing influences that are not apparent. Assessment of intermittent alarms should include consideration if similar behavior has been observed in the past (i.e., the alarm was observed previously without resolution). See also the discussion of intermittency in Section 3.4.

Problem: Managing False Alarms due to Transient or Intermittent Behavior

Recommendation: Transient or intermittent system behavior should be considered when detecting or interpreting alarms. Approaches to preventing false alarms due to transience include waiting to confirm steady state behavior or considering the likelihood of transience before acting on the alarm. When transient behavior is intermittent, it is necessary to provide information about previous instances of that behavior.

Example: The example in figure 4-32 illustrates false alarms caused by transient behavior. In figure 4-33, the intelligent system avoids generating a false alarm by waiting a few data cycles to determine if the effect is transient or permanent. This example is based on Scenario 1.

System misconfiguration can result in false alarms. Misconfigured devices may appear to be failed devices because they do not exhibit behavior consistent with the expected configuration (e.g., default value at power-up may differ from desired default for a specific mission phase). In the development of the IESP, it was observed that misconfigurations are especially common after a transition to redundant capability. This was true because many devices power-up in a configuration other than the nominal operational configuration. False alarms due to misconfiguration can be minimized by having the intelligent system compare the configuration of a device exhibiting anomalous behavior with the expected device configuration consistent with the current mission objectives before declaring a failure.

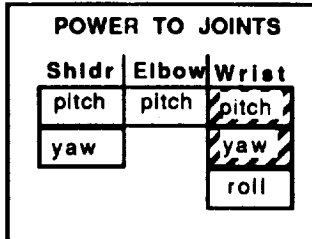
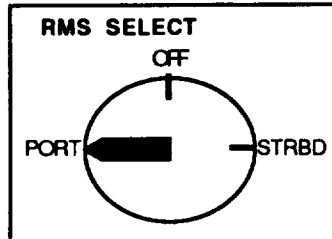
Problem: Differentiating Misconfiguration from Failure

Recommendation: The intelligent system should be able to differentiate system misconfigurations from system failures. Presentation of information about such system anomalies should clearly distinguish these two situations.

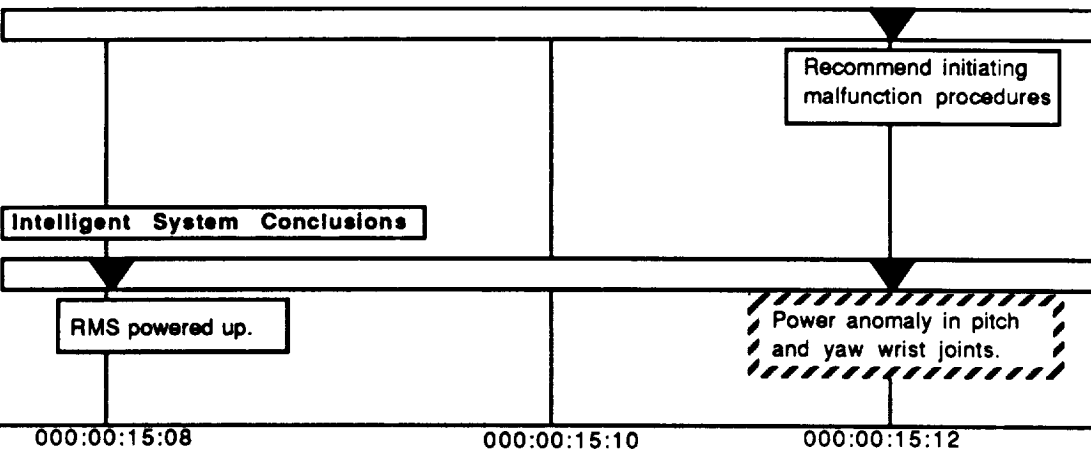
Problem: Managing False Alarms due to Transient or Intermittent Behavior

BEFORE - Example Illustrating Problem:

In this example, the intelligent system makes erroneous recommendations based on a false alarm. This alarm resulted from a transient signal at powerup of the RMS.



Intelligent System Recommendations



STEP 1
RMS is powered up by setting the RMS select switch to PORT (i.e., RMS on left side of vehicle).

STEP 2
False alarm is issued in both pitch and yaw joints of wrist. Alarm indicates no power received in these joints and is caused by a time delay in receiving the power signal. Delay results from low temperatures.

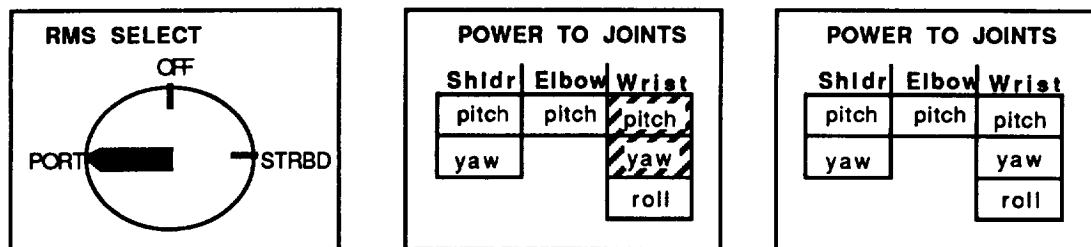
STEP 3
Intelligent system does not know that this is a false alarm and recommends malfunction procedures to identify source of power problem.

Figure 4-32. BEFORE: Example Illustrating False Alarms caused by Transient Behavior

Problem: Managing False Alarms due to Transient or Intermittent Behavior

AFTER - Example Illustrating Solution:

In this example, the intelligent system recognizes that false alarms often occur at powerup due to transients. No action is taken on the initial alarm and the intelligent system waits for a few data cycles before assessing the situation. After a few cycles, the transient disappears and the alarm goes away. The intelligent system correctly assesses a nominal powerup.



Intelligent System Conclusions

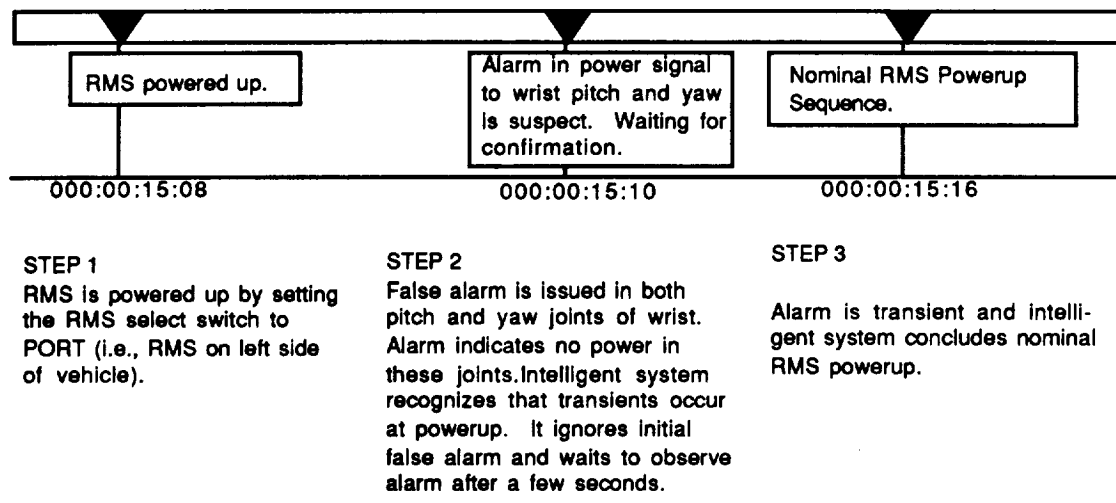


Figure 4-33. AFTER: Example Illustrating Design that Avoids False Alarms caused by Transient Behavior

Redundant Alarms

Redundancy is a common form of risk management within aerospace applications. Redundant alarms are two or more independent alarms that provide notification of the same anomalous condition. Redundancy may be provided as a backup in case of alarm failure. It may also result indirectly, when a failure in one component causes an alarm state in a related component (e.g., loss of power may disable a fan used for cooling, causing a rise in temperature in the cooled item). Redundant alarms reduce the risk of a false alarm by providing multiple, independent indicators of a failure. When redundant alarms provide inconsistent information, however, the decision of which alarm to believe becomes an issue. Common methods for resolving alarm inconsistency include accepting the consensus of the majority or a weighted combination of alarms. For the second approach, alarms are weighted because some alarms are considered more reliable than others. See section 4.3.1.1 for a discussion of redundant information sources.

Redundant alarms can contribute to operator workload. The operator must combine multiple information items into a single piece of diagnostic information. Synthesis of redundant alarm information involves identifying related alarms, resolving alarm inconsistency, and associating alarms with a failure. Sequence of alarm annunciation can also indicate the failure causing the alarms.

Problem: Relieving Operator Overload due to Redundant Alarms

Recommendation: The intelligent system should assist the operator in managing redundant alarms to emphasize the diagnostic content of the alarms and to avoid overloading the operator with duplicate information. Techniques for managing alarms include combining alarm information, using redundancy to establish confidence, and using presentation of information to associate related alarms.

Possible techniques for management of redundant alarms include:

- Combining redundant alarms into a single information item (i.e., a composite alarm)
A potential risk to this approach is the loss of additional information provided by the redundant alarms, such as certainty of failure when all alarms are activated.
- Associating a confidence measure of the accuracy of an alarm based on agreement within the redundant set
A candidate for this measure is the percentage of the redundant set that is consistent (e.g., 2 out of 3 indicate a failure, therefore a failure is likely).
- Using presentation of alarm information to associate redundant or related alarms
For example, redundant alarms can be physically grouped on the display or organized where distinctive, recognizable patterns are formed when they are active.

Figure 4-31 from the discussion of false alarms illustrates a graphical technique for presenting information from redundant sources for easy comparison of value (e.g., use of a histogram for comparing measurements from sensor 1 and sensor 2).

4.2.2 Critical Diagnostic Information

Once the presence of an anomaly has been alerted by system alarms, the fault management team must initiate diagnosis to identify and rectify faults causing the anomalous behavior. The information essential to diagnose faults in the monitored process is discussed in this section. Recommendations address such issues as maintaining operator awareness of situation, comparing predicted and actual state sequences, evaluating consequences and alternatives, monitoring procedure execution, assessing the impacts of procedures, determining the impacts of failures, assessing the functional capability remaining after a failure, and responding to unanticipated situations.

Visibility into Monitored Process

Assessment of a situation using data from the monitored process was identified as a necessary activity for fault management in section 3.3.1. The intelligent system should assist the operator in achieving and maintaining an understanding of the current fault management situation, especially in situations of change, risk, and uncertainty that often accompany system failures. The types of information that provide visibility into the monitored process include changes to expected system configuration, existing failure conditions (especially those with significant impacts), relevant events affecting the monitored process and the environment, and summaries of fault histories (Johns, 1990) and agent activities affecting the monitored process. Since operator's awareness of the current situation is dependent on the availability of information from the monitored process and its peripheral systems, such awareness can be compromised during loss of data, either partial or complete (see section 4.3.1.1 on information availability).

The effectiveness of information in describing the current situation is also dependent on the methods of information presentation. These methods should focus operator attention on information that is diagnostically important and should quickly and clearly illustrate the diagnostic content. Information should be presented within the context of the events and agent activities leading up to the current situation. To illustrate dependencies between events and activities, such information should be displayed chronologically. See also section 4.1.1 for a discussion of situational assessment at operator handover and section 4.2.1 for alarm management.

Problem: Maintaining Awareness of Monitored Process Situation

Recommendation: Information describing the current situation in the monitored process should be presented within the context of events and agent activities preceding the situation. Information that should be provided includes histories of system configuration changes, relevant events affecting the system and the environment, and agent activities in response to those events. It may be important to represent this information in chronological sequence, since the order of events affects their interpretation.

The upgrade to a new or different technology often alters operations. This change in operations can inadvertently remove access to information critical to the operator's situational awareness. Information useful in clarifying the current situation can be acquired informally as a by-product of current operations (Buck, 1989). Norman (1990) cited an example of information loss due to a technology upgrade in commercial air traffic control. Airline voice communications were replaced with electronic communications. The pilot's awareness of the current situation was impacted by the loss of incidental information acquired by listening to

other conversations on the voice loop. This is a potential problem for space flight control as well.

An important capability for situational awareness is the ability to review information from the past. It can be an orienting technique for in-coming operators at shift handovers. It is a means of evaluating behavior trends. It can also be an important mechanism for reminding an operator of important occurrences. See section 4.1.2 for a discussion of reviewing information with an intelligent system.

Evaluation of Consequences

The consequences of an event are evaluated by predicting the effects of the event on the system of interest and determining if these effects impact either safety or mission goals. The consequences of an anomaly include the propagated effect of the anomaly (e.g., predicted effect of a fault over time if not corrected) and the risks associated with a planned response to the anomaly (e.g., effect of intervening into intelligent system processing). This evaluation assists in selecting between alternative solutions (e.g., compare the effects of alternative workaround procedures) and in setting activity priorities used when scheduling anomaly response (e.g., schedule recovery procedures for high risk failures first).

Problem: Support for Evaluating Consequences of Events and Team Activities

Recommendation: The operator should be able to evaluate the expected consequences of changes resulting from anomalies. Changes affecting the monitored process include failure effects propagated over time and the impacts of executing procedures for fault testing, workaround, and recovery. Changes affecting the intelligent system include operator intervention into system processing.

WHAT-IF analysis is a common method of evaluating consequences that involves both the human and the intelligent system. The name is derived from the question "What would happen if ...". WHAT-IF analysis consists of postulating a change in the current situation (e.g., a fault, execution of a procedure) and determining the resulting impact to system capability. WHAT-IF analysis is not considered to be a real-time operation and any display regions designated to present the results of this analysis should not be updated with data in real time. Typically, upon initiating WHAT-IF analysis, the current configuration of the system in question is frozen and no further updates from real-time data are reflected in the on-going analysis. All further changes to the system are initiated by the collaborating agents as a part of investigating the potential effects of hypothesized faults. An example of WHAT-IF analysis was seen in the GNC Jet Control system (see Volume 2, Malin et al., 1991).

Display of the results of the evaluation of consequences is a variant of the display of expectations and predicted results. See the discussions of predicted behavior, alternative states and behavior, and impact assessment later in this section. See section 4.2.3 for evaluating the consequences of workaround and section 4.1.3 for the consequences of operator intervention into intelligent system processing.

Predicted Behavior

In section 3.2.3, simulation was one of the methods described for generating information. One important use of simulation is prediction. The prediction of state values for the monitored process provides information important in evaluating the consequences of an event or planned

activity. Predicted behavior can be compared to actual behavior to determine if planned activities cause the desired effect when executed (e.g., procedures monitoring) or to detect when the current situation does not agree with the expected situation based on on-going operations (e.g., detect anomalous, actual behavior).

A *prediction sequence* is the state sequence describing expected behavior of the monitored process (Johns, 1990). Johns discusses a number of issues associated with comparing predicted behavior to actual behavior including identification of the important events (i.e., *milestones*) where a comparison should occur, definition of criteria for identifying when behavior agrees and when it does not, and definition of divergence in terms of milestones not satisfied.

A part of comparing predicted and actual behavior is guaranteeing that the assumptions used to predict behavior match the current situation. When these assumptions are no longer applicable, the comparison becomes invalid. Thus, a description of expected behavior should include assumptions defining conditions when the predicted behavior is possible (e.g., the Space Shuttle startracker sensor can only track a satellite when the sun is visible, since tracking is based on light reflected from the satellite). These conditions can be expressed as time constraints or event dependencies. The specification of milestone behavior should include conditions required for the behavior to occur.

Since future conditions can alter the predicted behavior, there can be more than one possible description of future behavior. The term used in qualitative simulation to describe this range of possible future behavior is *envisionment*. Envisionments can rapidly become multi-path and strongly interrelated in complex environments like aerospace. Effective display of envisionments is an unresolved issue requiring further investigation (Forbus, 1991).

Issue: The display of expectations and predicted results is an area requiring further investigation. Issues include:

- Simultaneous display of information resulting from different environmental and configuration assumptions
- Definition of what constitutes agreement between information from multiple sources (i.e., when do they match)
- Definition and display of divergence and criticality of divergence (i.e., Is this an important mismatch?)
- Display and control of predictive simulation.

Display of multiple futures (i.e., envisionment) is also important. See also in this section the discussion of alternative states and behavior, evaluation of consequences, and the display of expectations based on procedures. See section 4.3.1 under quality of information for discussion of inconsistency and significance of disagreement when comparing information items.

Alternative States and Behavior

Consider managing a Space Station thermal control system (TCS). Suppose there is a small coolant leak which will require Extra-Vehicular Activity (EVA) to fix. If a maintenance EVA is already scheduled just after the next re-supply ship, one option is to simply wait until after the next Space Shuttle arrives. If the leak is not serious, and it is possible to monitor the TCS more closely to get early warning if the rate of coolant loss increases substantially, the waiting strategy looks even more plausible. At this point the team might go over the current and planned onboard operations, to see if thermally expensive activities could be terminated or rescheduled to increase the safety margin. It could be the costs of doing so are too high -- termination of expensive experiments, for instance, or the thermal load simply being large

enough that the safety margin is too small, making an extra EVA the preferred alternative. Or it might be that changing a few activities around will keep the thermal load small enough that it makes more sense to wait.

This scenario illustrates a key feature of many fault management problems: the need to reason about alternative states and behaviors. Given that the coolant is periodically replenished, in fact the amount of coolant may never get so low that heat transfer capacity is threatened. But in fact it might, and detecting this possibility should cause attention to focus on resolving this ambiguity. Once the relevant durations and likelihoods have been found, the team must generate and examine alternate strategies for dealing with the situation. The intelligent system must be able to "stand back" and view several distinct alternatives at once in order to fully participate in the deliberations. We do not know of any applications-oriented efforts to date which maintain and reason about branching time.

The lack of detail in qualitative representations is what gives rise to ambiguities in predicted behavior in qualitative simulations. Since traditional simulations yield unique predictions of behaviors, it is tempting to suspect that if we eschewed qualitative representations we could avoid thinking about branching time. This is not the case. Realistically, ambiguity typically exists even with quantitative data. Sensors provide noise as well as information about a physical system's state. System states can only be estimated with limited accuracy. Models have limited accuracy, or may not even be available if unusual and/or unanticipated failure conditions arise. Ergo, the ambiguities of qualitative reasoning may be the only information available.

It should be noted that the ability to maintain alternate interpretations of the past can be just as important as maintaining alternate interpretations of the future. In monitoring, for instance, sometimes a current state which is slightly off may only indicate a small excursion in external conditions which is still being compensated for. But it could also indicate that a failure has occurred in a system which is not directly sensed, and worse changes are yet to come. If such changes occur, dealing with them could require going back to old sensor records and examining alternate interpretations which could explain the current circumstances better.

It seems there are several orthogonal factors in communications about a physical system's prospects. First, one must describe the classes of behavior which might occur. To a first approximation the normal notion of qualitative state and transition may suffice. However, for describing the gist of a problem concerning a complex system it will be important to work with suitable abstractions of the raw dynamics. The second factor is the likelihood of each projected path of behavior. A third factor is representing the evaluation of each projection relative to the performance goals of the system. A fourth factor is the temporal proximity of each projected change. The fifth (and so far final) factor is describing any interconnectedness between alternate projections (e.g., joins in the envisionment graph).

The size of this list may seem surprising at first, but hopefully it can impose some order on many concepts in monitoring and reasoning about system operations. For example, "urgency" is a slippery concept. Intuitively, judgments of urgency seem context-sensitive: The potential for corrosion wearing out a pump before the next scheduled maintenance typically drops in urgency when a coolant leak is detected. Perhaps by viewing urgency as a functional combination of evaluation, likelihood, and temporal proximity information, we can provide a better handle on such concepts.

Issue: Techniques for displaying alternate possible futures and pasts are needed for fault management. Factors that should be considered when communicating about alternative states and behaviors are:

- Classes of behavior which might occur
- Likelihood of each projected path of behavior
- Evaluation of each projection relative to the performance goals of the system
- Temporal proximity of each projected change
- Interconnectedness between alternate projections

Finding good ways to display alternate possible behaviors is a hard problem. The problem is that behaviors are not simple sequences, or even trees -- currently distinct options can eventually lead to the same behavior, and oscillation gives rise to cycles in qualitative states. Thus one is faced with displaying general graphs, which is substantially harder than displaying trees.

Displays used in qualitative physics research have used abstract icons (e.g., circles or squares) for states and indicated transitions between them with arrows. One could certainly improve this, by for example using state icons which indicated more information about that state (such as what processes are occurring). Perhaps the biggest gain is in task-specific layout strategies. Consider the display of possible futures a monitor might generate. It needs to get across the likelihood, relevance, and temporal proximity of these alternate behaviors. Suppose furthermore it has quantized each of these three factors. Then we could (1) sort states into clusters based on equivalence relations in these factors, (2) lay out each cluster to make the transitions as clear as possible, and (3) arrange the clusters to make important properties of the envisionment visually apparent. For example, temporal proximity could be classified by sorting states into whether their projected occurrence is minutes, hours, days, weeks, or even months away from the hypothesized current state. Imagine now a grid whose vertical coordinate consists of equivalence classes of estimated likelihood and whose horizontal coordinate consists of equivalence classes of temporal proximity. States closest to the upper-left corner of the display, for instance, would be the ones to attend to first, since they are both most likely and soonest to arrive. Long-term prospects would appear on the right, with the least-likely prospects appearing on the bottom. If some other visual property, say color, indicated relevance (e.g., nominal versus unsafe versus dangerous), then this display might substantially simplify communicating priorities.

Mission Impacts and Procedures

Procedures are plan elements specifying sequences of agent activities to achieve mission goals (section 3.3.1). The execution of a procedure can impact the monitored process or the peripheral systems that influence the monitored process. Planned or intentional impacts are the means of accomplishing mission goals. Unplanned and secondary impacts (i.e., side-effects) are also possible, and introduce the prospect of negative effects and unanticipated situations.

The relationship between procedures and their impacts (planned and unplanned) is important in managing the monitored process. The fault management team must be able to predict the expected effects of a procedure, monitor the execution of the procedure by comparing those expectations to actual behavior, alert unexpected or harmful impacts, and determine new procedures in response to negative impacts (section 4.2.3). The intelligent system should assist the operator in performing these activities. Issues affecting human-computer interaction for procedures monitoring and execution include displaying expectations (see the previous discussion of predicted behavior), illustrating the relationships between procedures and their impacts, and alerting unplanned, incidental impacts of procedures.

Problem: Support for Evaluating Impacts of Procedure Execution

Recommendation: To assist the operator in monitoring procedure execution, the intelligent system should support such activities as predicting the expected effects of a procedure, comparing expected behavior to actual behavior, alerting unexpected or harmful impacts, and determining new procedures in response to negative impacts. Information required for these activities includes procedures, scheduled activities, potential impacts of procedures, and criticality of impacts.

The resolution of these human-computer interaction issues must address not only information presentation but must identify the information requirements for procedures monitoring and impact assessment. Mission goals must be related to scheduled procedures. Changes to scheduled procedures must be evaluated for impact to mission goals and safety. These potential impacts should be accompanied by warnings if those impacts are irreversible (e.g., preclude the ability to determine critical information) or fail to meet flight rules. The significance of an impact to on-going (and possibly unrelated) activities should also be evaluated. The fault management team must be able to determine when an impact is of minimal consequence and when it is potentially harmful in a given situation. The fault management team must recommend modifications to crew activities only after evaluating the net impact of these changes.

Issue: Information requirements and effective presentation techniques for monitoring the execution of procedures and assessing their impact require further investigation. Areas for study include:

- Displaying procedures and activity sequences
- Displaying expectations
- Distinguishing expectation from actuality
- Relating procedures to their impacts on the monitored process and on other activities
- Alerting unexpected impacts of procedures
- Differentiating between a significant impact and an insignificant impact

A useful related capability is monitoring the execution of these procedures and displaying the results of their execution with respect to the expected results and potential impacts. One method of determining potential impacts is to perform a WHAT-IF evaluation (see the previous discussion of evaluation of consequences).

The monitoring of procedures was performed by some of the applications in the case study, including the KU Band Self Test Expert System, the OMA Prototypes, REX, and the GNC Real-time Monitor. See Volume 2 (Malin et al., 1991) for a description of these systems.

A procedure normally consists of multiple activities in an ordered execution sequence. One goal of procedures monitoring is to ensure that all activities are executed in the proper order. In some cases, the procedure will include activities performed conditionally based on the circumstances at the time of execution. Thus, the expected activity sequence must include multiple possible sequences. The types of information useful for monitoring the sequence of procedures should be identified.

Issue: The information required to monitor procedure sequence should be defined. These information requirements should support the following fault management activities:

- Illustrate when order is critical to the success of the procedure
- Alert out-of-order activities
- Assess the impact of out-of-order activities and suggest ways to correct this impact, if possible
- Display procedures with conditional activities that result in multiple possible activity sequences

A timeline of activities is one technique for illustrating activity sequence. An example of the use of timelines to illustrate events and activities was seen in the PDRS HCI design concepts (Volume 2, Malin et al., 1991).

Functional Capability Assessment

Procedures produce intentional impacts in the monitored process. Anomalies can introduce unintended impacts in the monitored process by impairing functionality required for scheduled operations. In addition to impacts manifested immediately, the potential for an anomaly to impact the monitored process at some time in the future may also exist due to failure propagation over time. When a failure occurs, the resulting loss of functionality in the monitored system must be determined, the remaining functional capability assessed, and the potential for the failure to propagate must be predicted.

To assess functional capability loss, failures in the monitored process must be identified. Johns has defined the *failure summary group* as a list of all failures in the monitored process organized by the associated problem (Johns, 1990). Associated with a failure group is an explanation of the anomaly and supporting information, which can include graphic portrayals. The failure summary group is a useful way of representing failure information for functional capability assessment.

Before the impacts of an anomaly can be determined, the behavior that represents an anomaly must be defined. Normal behavior for a system can vary from the design specification of behavior (e.g., a component may operate normally at a temperature slightly outside design specifications or a component's operating characteristics may change over time). Although such variation represents no loss of capability, it can violate design criteria for expected behavior. These design criteria represent an envelope of expected behavior that should be explicitly defined with the system design. This envelope should be modifiable, to allow changes to a more accurate representation as operational experience is gained. This is particularly important for a newly designed monitored process (such as systems on the Space Station) where the operating characteristics are not well known.

Examples of monitoring for behavior outside an envelope of expected behavior were observed in the case study. Two systems, the KU Band Self Test Expert System and the GNC Real-time Monitor, monitor pre-specified parameters for values outside normal operating limits during check-out tests of specific subsystems. See section 4.2.1 for a related discussion on managing false alarms when monitoring parameters.

Not only must the boundaries of nominal behavior be defined, but the severity of anomalies must be determined. This is related to the criticality of the functional impacts resulting from the anomaly (i.e., When is a deviation out of bounds but of minimal impact and when is it out of bounds with potentially harmful impact?). Also, once an anomaly has occurred, the definition of expected behavior may change (what Johns (1990) terms *expectable* behavior). For example, when a light bulb fails on the Display and Control Panel, expected behavior for the associated panel indicators would be redefined to "always off" until the bulb can be replaced.

The Loss of Control application (Volume 2, Malin et al., 1991) for the GNC flight control position defines regimes of behavior when monitoring for anomalies in the Space Shuttle attitude indicating loss of vehicle control. These regimes of behavior correspond to levels of severity in anomalous behavior of the vehicle control system. See table 4-7 for an illustration of these levels of severity.

Table 4-7. Levels of Severity for Space Shuttle Vehicle Control Anomalies

Regimes of Behavior for Space Shuttle Attitude	
• Normal	Attitude angles are changing at a normal rate
• Going Out of Bounds	Attitude angles are changing at a rate greater than expected; potential to lose control of vehicle
• Loss of Vehicle Control	Attitude angles indicate vehicle is spinning out of control

For Space Shuttle, Critical Items Lists (CILs) identify the functional capabilities and hardware systems critical to crew safety and mission success (see appendix D.2). A critical item is a system or capability with a single failure point (JSC, January 1989). Redundant elements can be critical items in a life- or mission-essential application where redundancy cannot be checked out, loss of redundancy is not apparent, or a single event can remove all redundancy (JSC, January 1989). Critical items and the availability of redundant capability should be identified. There is a need to identify how to effectively present this information

Problem: Support for Determining Remaining Functionality after Failure

Recommendation: To assist the operator in assessing the functional capability remaining after a failure in the monitored process, the intelligent system should support such activities as determining the loss of functionality in the monitored system, assessing the remaining functional capability, and predicting the potential for the failure to propagate into other systems.

Issue: Investigation of the information requirements and display techniques for functional capabilities assessment is needed. This issue is related to the functional impacts of a failure, criticality of functions, and failure propagation potential. The relationship between the availability of redundant functionality and the associated criticality rating of that functionality also affects functional capabilities assessment.

Failure propagation potential includes assessing the expected impacts of a failure and the ways that the failure could propagate into other failures. Failure propagation can occur within a

specific system or among different systems. Functional dependencies can exist between systems (e.g., loss of a power bus can shutdown a fan cooling a system and result in overheating). The existence of such dependencies associated with a failure can result in global failure effects, including failure propagation across multiple systems.

Issue: Methods are needed for representing functional dependencies among different systems and assessing the resulting global effects and functional impacts of such failures.

WHAT-IF evaluation can be used to determine failure propagation potential. If such a formulation is used, see the previous discussions of the evaluation of consequences, predicted behavior, alternative states and behaviors, and mission impacts and procedures for display of functional impacts resulting from procedures execution.

4.2.3 Unanticipated Situations and Workaround

Procedures define the behavior expected from the monitored process and the activities of agents participating in fault management during situations that are anticipated prior to a mission. In actual operations, contingency situations arise where the behavior of the monitored process or the intelligent system does not match these expectations. Such situations arise from a variety of causes, including operator misentry, misinterpretation of situation, inherent variability of devices, software errors, or the occurrence of multiple failures (Woods, Roth, and Bennett, 1990). Since a contingency situation has not been anticipated, no procedures exist to respond to the situation. One of the difficulties in an unanticipated situation is that the problem is ill-defined or unknown. The operator must first problem-solve to identify the problem, then problem-solve to correct the problem (Lemke and Fischer, 1990). The operator should be able to manipulate information to identify the source of the contingency and generate and test workaround procedures in response to such contingencies. Workaround procedures are procedures developed in real time in response to contingency situations. See also section 3.3.1 for a discussion of re-plan at contingency situations.

To respond to contingency situations and unexpected anomalies, the operator must have access to mission plans, such as mission procedures and scheduled activities. The intelligent system can assist in an analysis of the resulting workaround procedures by providing such information as the prerequisites for the execution of the procedure, the potential consequences of executing the procedure in the given situation, violations of flight rules caused by the procedure, and necessary post conditions for the altered procedure to succeed (Woods, 1986). Once these procedures have been initiated, the operator should be able to monitor the effects of the workaround procedures on the anomaly. See section 4.2.2 on monitoring procedure execution.

Problem: Support for Developing Workaround Procedures

Recommendation: The intelligent system should be designed to assist the operator in handling unanticipated situations. The operator should be able to generate and test workaround procedures. Available information should include pre-defined procedures, assessment of procedure impacts, scheduled activities, and flight rules.

For example, an analysis of procedure impacts is very useful during the construction of workaround procedures. The execution of a procedure can impact crew and vehicle safety, mission goals, and on-going operations. It can even affect the ability to perform fault management by altering the diagnostic situation (e.g., change data from the value at system

failure). The trade-offs between the importance of executing a procedure and the negative impacts of that procedure are an important part of revising planned operations to respond to anomalies. See section 4.2.2 for discussions of procedure impacts and evaluating the consequences of events and activities, including execution of workaround procedures.

Issue: Study is needed to determine complete information requirements and effective display techniques to assist the operator in handling unanticipated situations. Areas for further study include:

- Detection of atypical situations
- Machine compensation for human stress reactions
- Display of ambiguous problem space
- Display of uncertainty (e.g., what do I know, what might be true; section 4.3.1.1)
- Display of response options with impacts
- Criticality of response options (e.g., potential impacts if NO response, failure propagation; section 4.2.2)
- Operator orientation in presence of interruptions (section 4.1.1)
- Managing multiple failures (e.g., failure priority and criticality, failure dependencies)
- Emphasizing what has changed from what remained the same

Workaround operations cannot be planned and tested as thoroughly as pre-defined procedures. Such activities should be approached with caution due to the inherent risks of altering pre-defined activity sequences. In such a case, the solution could create worse impacts to safety or mission than the original anomaly. Any capability associated with workaround should include the ability to evaluate the risks and potential impacts of that workaround.

Problem: Minimizing Risk Introduced by Workaround Procedures

Recommendation: To minimize the risk inherent in real-time changes to procedures, the ability to evaluate the potential impact of workaround procedures, including changes in agent activities and reallocations of responsibilities, should be provided.

Thus, the ability to perform dynamic workaround must be balanced against the resulting risks of such workaround and the requirements of configuration control. Since the time to accomplish a workaround is frequently limited, performance constraints must also be considered. These factors are a major constraint on what type of control and how much control is provided to the operator in unanticipated situations. In the eventuality that the unanticipated situation involves loss of capability in the intelligent system, the operator may need to intervene in the machine reasoning process (section 4.1.3). In an extreme case, the operator may have to takeover from the intelligent system.

The additional information required to support workaround can have a cost. More information to manage can increase operator work load. The system should be designed to assist the operator in managing the larger amounts of information required to support workaround. Sections 4.3 and 5 provides recommendations for managing information overload.

For additional information related to unanticipated situations, see the previous discussions on critical diagnostic information in section 4.2.2. See section 4.1.1 for recommendations concerning agent task reallocation in response to unanticipated situations and interruption.

4.3 Support for Information Management and Display

Information management is the manipulation of information to assist in interpretation and use of information. Due to the potential for overloading the operator with large amounts of dynamic information during real-time fault management, it is important to assist the operator in managing information. One aspect of managing information is the ability to interpret information. Section 4.3.1 provides recommendations for designing to assist interpretation of information.

Woods has introduced the term *workspace* to describe the screen real estate provided by the user interface and the set of all graphics and text displayed within that real estate (section 5.1.2). Design of the workspace affects the ability of the operator to access and manipulate information effectively. Section 4.3.2 provides recommendations and issues addressing design of the workspace for information management.

4.3.1 Interpretation of Information

Managing information overload requires designing the intelligent system and its user interface to assist the operator in interpretation of information. Information context can be used to present information in a more meaningful fashion. Qualitative representation of information can be an effective way to identify information content. To effectively manage information overload, information presentation should go beyond visualization (which still requires a substantial operator effort to interpret) to support summarization of information and suppression of irrelevant detail. These are areas requiring further investigation.

4.3.1.1 Information Context

In section 3.2.3, characteristics common to all information are identified as the source of information, quality of information, and availability of information. These information attributes represent a context useful in interpreting an information item. In this section, recommendations and issues associated with communicating these attributes of information to agents of the fault management team are discussed.

Source of Information

It becomes important to identify the source of an information item when information from multiple sources is viewed from one workspace. Knowledge of the source of an information item identifies how the information was generated and clarifies the authority of that item. Understanding the method of generation assists in interpreting information. The source identifier provides a reference for making related information requests (e.g., query an operator about details of an input). It is useful for quickly identifying when backup or redundant capability is being used (Johns, 1990). Knowledge of the source of information items also assists in orienting an operator coming on shift about on-going events and the current configuration of systems. See section 3.2.3 for more about the source of an information item.

Problem: Determining the Authority and Credibility of Information

Recommendation: Agents should understand the source of information used in fault management. The source of information identifies how the information was determined and indicates the authority of the information. It is especially important to identify the source of derived (i.e., computed and inferred) information, since the operator must understand the algorithm or reasoning behind the information to be able to interpret the information.

Example: The example in figure 4-34 illustrates operator confusion arising he cannot distinguish between information from the intelligent system and information from another operator. In figure 4-35, the source of information is clearly indicated on the display, allowing the operator to assess information reliability based on its source. This example is not based on a scenario. The background required to understand the example is provided in the figures.

There are a variety of techniques for indicating the source of information on the user interface. Text identifiers can be placed near the information item (e.g., a field for source identifier in message list, character identifier adjacent to data field). Graphic forms can also be used to identify information source (e.g., icons can be uniquely associated with specific sources). Coding techniques (e.g., color, highlighting) can also be used when displaying information. Color should be used conservatively, since color overuse is a common problem. Although it may not be necessary to always have source visible, the operator should be provided with ready access to source information.

One of the characteristics related to the source of information is the availability of alternate or redundant sources. The authority of a source is affected by the degree of confidence that the operator has in the source. When alternate sources are operating nominally and providing information, the agreement between these sources about the information item directly impacts confidence. Since some sources may be considered more reliable by the operator, the identity of the selected source also affects operator confidence. Thus, the availability and health of alternate sources and the identity of the selected source when alternates exist affect the authority of the information item. These attributes of an information item should be accessible by the operator.

When displaying information from alternate sources on one screen, it is important to clearly identify differences between the alternate sources. Often, the difference is the method used to generate the information. The comparison of a predicted or simulated information item to the corresponding real information sampled in the actual environment (e.g., comparing a predicted measurement to an actual measurement) is a common situation where the method of generation is the critical source information. See section 4.2.2 for a discussion of predicted behavior.

If alternate sources are simultaneously active, the potential for conflicting information exists (e.g., hardware fault may be detected both by the Built-In Test Equipment (BITE) and by the intelligent system diagnosis). See the discussion of inconsistent information under quality of information later in this section for methods to evaluate and resolve inconsistency.

Problem: Determining the Authority and Credibility of Information

BEFORE - Example Illustrating Problem:

Knowledge of the source of information often indicates the believability of the information. In this example, source of information cannot be distinguished from the display. The operator has difficulty in determining how reliable a piece of information is, because he does not know how it was derived.

EVENT SUMMARY	
103:09:13:00	The Port MRL has failed to release.
103:09:12:42	The Port MRL Release commanded.
103:09:11:36	DAP in free drift.
103:09:10:20	The MPM has deployed successfully.
103:09:09:10	Expect Single Motor Drive Time.
103:09:09:08	MPM DEPLOY commanded.
103:09:08:50	Limit check port aft MRL microswitch REL, MSID V54X2345

DIAGRAM 1

The operator cannot determine which events were detected by the intelligent system and which events were detected by a previous operator using this message list.

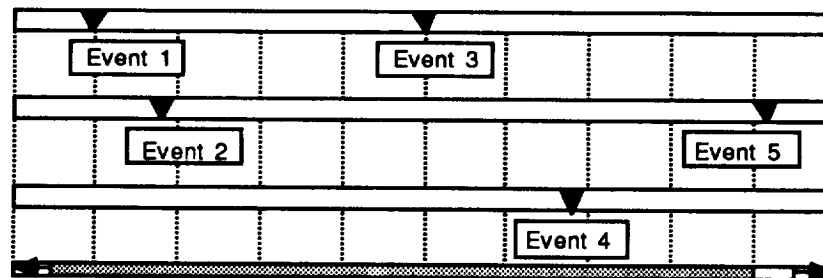


DIAGRAM 2

Similar to diagram 1, the operator has no means of identifying the source of an event in this timeline representation of events.

Figure 4-34. BEFORE: Example Illustrating Confusion about Source of Information

Problem: Determining the Authority and Credibility of Information

AFTER - Example Illustrating Solution:

In this example, source identifiers are added to the entries in a message list and a timeline. The current operator can now distinguish between conclusions of the intelligent system and information provided by previous operators.

EVENT SUMMARY	
103:09:13:00	RULES The Port MRL has failed to release.
103:09:12:42	RULES The Port MRL Release commanded.
103:09:11:36	COMPS DAP in free drift.
103:09:10:20	RULES The MPM has deployed successfully.
103:09:09:10	RULES Expect Single Motor Drive Time.
103:09:09:08	RULES MPM DEPLOY commanded.
103:09:08:50	DCULP Limit check port aft MRL microswitch REL, MSID V54X2345

DIAGRAM 1

The second field of each message contains a source identifier (e.g., RULES, COMPS). Sources in this illustration include the intelligent system (RULES), conventional algorithm (COMP), and the operator responsible for entering information (e.g., DCULP for operator Don Culp).

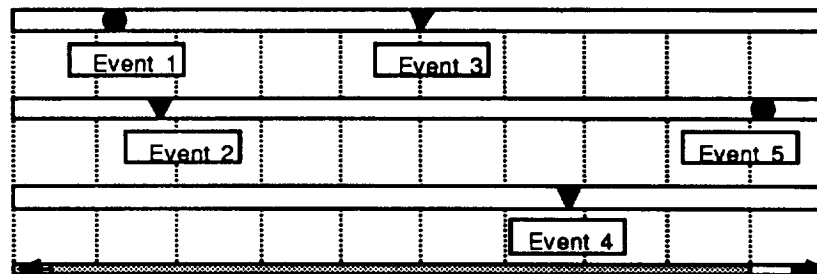


DIAGRAM 2

Graphic forms can also be used to identify information source. In the event timeline above, the triangular mark indicates an event detected by the intelligent system and the circular mark indicates an event detected by the operator.

Figure 4-35. AFTER: Example Illustrating Clear Identification of Source of Information

Problem: Resolving Disagreement between Redundant Sources of Information

Recommendation: Information useful in distinguishing between alternate information sources includes availability of information from the source, the health of the source, and the reliability of source (as indicated by previous behavior). Measures of consistency of information from different sources should be provided and methods for resolving inconsistency. When redundant sources are available, the currently selected or enabled source should be identified.

Issue: Techniques are needed for effective display of information from multiple, alternate sources, including redundant sources.

Figure 4-31 in section 4.2.1 includes an illustration of redundant sensor measurements when they disagree. Figure 4-37 in the section below (quality of information) provides another example of presenting information from alternate sources (e.g., quality assessments from BITE and intelligent system). See also section 4.2.1 for information concerning redundant alarms.

The operator can also be a source of information. Interaction to acquire information from the operator can be initiated by either the operator or the intelligent system. If initiated by the intelligent system, the information can either be requested or the information can be inferred and the operator requested to confirm it. The utility of these different mechanisms is dependent upon the context in which the interaction occurs (Johns, 1990). For example, the ability to infer information and merely request the operator to confirm the inference is more effective if the human and computer have been collaborating about a given topic, which provides the computer with a context for its inference.

Issue: Multiple mechanisms are possible for operator input into the intelligent system: (1) requested by machine, (2) inferred by machine and confirmed by user, and (3) volunteered by user. Research is needed to determine criteria for identifying appropriate/effective mechanisms for operator input in different human-computer collaboration scenarios.

The sources of information may vary over the lifetime of the system. The evolution to digital electronic forms of information (e.g., from paper or analog electronic) is a common situation where alternate or new sources of information are introduced. For example, the Space Shuttle program is currently undergoing a change to digital voice communications, which will make voice information available in an electronic format. Upgrading an existing system to include intelligent system technology represents the addition of a new data source. In either of these situations, the introduction of a new technology results in a new source of information for the fault management team. It is planned that Space Station will use an electronic format for much of the information that is in paper for Space Shuttle (e.g., procedures). This will provide more potential sources of information for Space Station and the identification of source will be especially important. See section 6.3 for additional guidance on the integration of intelligent system technology into an existing support environment.

Quality of Information

An important capability in managing information from both the intelligent system and the monitored process is the ability to identify and compensate for imperfections in information. There are a number of types of imperfection, including imprecise information, inconsistent information, and ambiguous information (section 3.2.3). The operator should be provided the

capability to detect and compensate for imperfections in information. See Section 4.2.1 for a discussion about managing false alarms arising from imperfect information.

There may be multiple assessments of the quality of an information item. Quality assessment may be performed at different points during transmission or processing of the information to satisfy specific objectives (e.g., a test using Built-In Test Equipment is conducted on orbit to indicate health of measuring device while a bit-error check is performed to detect errors introduced during transmission). When using an assessment of data quality, the objective of the assessment and the sources of imperfection evaluated in the assessment should be clearly identified. For example, bit-error checking indicates nothing about the accuracy of the data as measured. Additionally, if no assessment has been performed, the data quality value should be indeterminate. The use of nominal values for default values can also result in misinterpretation of information.

Problem: Determining Quality of Information

Recommendation: Information quality assessments should be provided to the fault management team when available. The objectives of the assessment and the sources of imperfection evaluated in the assessment should be clearly identified.

Example: The example in figure 4-36 illustrates poor presentation of information from alternate sources of quality assessment (e.g., BITE and intelligent system), while figure 4-37 illustrates an improved approach. This example is not based on a scenario. The figure provides all background information required to understand the illustration.

Imprecise information is inexact or inaccurate information. Inaccuracy can result from approximation or from error. Approximation arises from precision limitations in hardware or software used to process the information as well as the fidelity of the source that generates the information. There are a variety of sources of errors in information, including errors in hardware devices (measuring and processing), errors during information transmission, and errors in information processing (i.e., software errors).

Accuracy is an assessment of how well an information item agrees with the criteria defining its expected precision. Information satisfying this criteria are rated as accurate (i.e., *good*) while information exceeding this criteria are inaccurate (i.e., *bad*). An assessment of information accuracy cannot always yield an absolute status (i.e., good or bad) for an information item. Data quality indicators may provide conflicting assessments or the available indicators may be insufficient to assess quality. An intermediate assessment, such as *suspect* data quality, can be used to convey the possibility of an unconfirmed imperfection in the value of information. An information item can be suspect due to its relationship with confirmed bad information (e.g., if B is derived from A and A is bad, then B becomes suspect), previous behavior trends (e.g., item has exhibited bad data quality before), inconsistency with other indicators (e.g., a sensor whose measurements disagree with redundant sensors is suspect), and ambiguities preventing quality determination (e.g., A or B is bad, but unable to determine which is bad).

Some types of knowledge representations provide an assessment of information certainty (e.g., membership in a fuzzy set, statistical representation). These assessments may be useful in assessing confidence in the accuracy of the information. See also section 4.1.2 (visibility into intelligent system activities and reasoning) for guidance on distinguishing information with different levels of uncertainty (i.e., hypotheses from facts).

Problem: Determining Quality of Information

BEFORE - Example Illustrating Problem:

In this example, the status of three sensors is displayed. The value of this status can be altered by either the Built-In Test Equipment (BITE, quality check built into the sensor) or by the intelligent system. With this presentation format, it is not clear which of these quality tests has determined the value. It is also not clear how a conflict between these two would be resolved.

SENSOR 1	Good	Suspect	Bad
SENSOR 2	Good	Suspect	Bad
SENSOR 3	Good	Suspect	Bad

Figure 4-36. BEFORE: Example Illustrating Poor Presentation of Alternate Sources of Quality Assessment

Problem: Determining Quality of Information

AFTER - Example Illustrating Solution:

In this example, status values from from the BITE and the intelligent system are displayed in close proximity to each other. This allows the operator to distinguish between quality tests (since they check different symptoms to determine quality) and to detect conflicts between them (here, BITE has detected Bad status on sensor 1 not detected by intelligent system).

SENSOR 1	BITE	Good	Suspect	Bad
	Int Sys	Good	Suspect	Bad
SENSOR 2	BITE	Good	Suspect	Bad
	Int Sys	Good	Suspect	Bad
SENSOR 3	BITE	Good	Suspect	Bad
	Int Sys	Good	Suspect	Bad

Figure 4-37. AFTER: Example Illustrating Improved Presentation of Alternate Sources of Quality Assessment

Issue: Techniques are needed to assist the fault management team in managing imprecise information. These techniques should indicate the certainty associated with the quality assessment (e.g., confirmed bad versus suspected bad), the cause of the inaccuracy, and possible ways to compensate for the inaccuracy.

Inconsistency occurs when multiple information items provide conflicting interpretations about a single event or aspect of behavior. These information items can come from alternate sources at a given time or from a single item sampled over time (see section 4.2.1 on intermittent behavior). Redundant capability for failure backup is one way of achieving alternate sources of information. Dependencies between systems can also give rise to alternate sources of information (e.g., loss of system power can be indicated by temperature increases in cooled systems and loss of data from measuring devices attached to the power system as well as by low voltage readings).

Comparing information for consistency requires that criteria for agreement and disagreement be established (see related discussion on criteria for comparing predicted and actual behavior in section 4.2.2). It is also important to identify when the disagreement is significant (i.e., has an important impact to safety or operations).

Once inconsistency has been detected, it must be resolved by determining a single interpretation of the conflicting information. Methods for resolving inconsistency between redundant alarms were discussed in section 4.2.1. These methods include accepting the majority opinion or creating a composite opinion by a weighted combination of information, where the weights correspond to the reliability of each information source. Statistical evaluation of information can also be useful in detecting transients or behavior trends that lead to imperfections (both inconsistency and inaccuracy) as well as selecting between alternate sources. The resolution should also be consistent with other system behavior (e.g., don't postulate a power loss if there is evidence that the device is still operating).

Issue: Techniques are needed to assist the fault management team in managing inconsistent information. These techniques should indicate the source of the information items being compared, the intent/objective of the comparison, the criteria for agreement, and the significance of a disagreement. Methods for resolving inconsistency should be provided also.

Ambiguous information is information supporting multiple interpretations. Ambiguity should be clearly indicated to the operator. The information needed to evaluate ambiguous information includes the set of possible interpretations of the information, the activities required to reduce ambiguity, and an assessment of the impacts of performing these activities. This impact assessment should indicate any violation of the mandatory operational conditions and constraints (e.g., the flight rules for Space Shuttle). Since the set of possible interpretations may change as activities are performed to reduce ambiguity, a display of ambiguous information should dynamically change to reflect the altered situation. See also section 4.2.2 for recommendations relating to procedures monitoring and impact assessment.

Issue: Techniques are needed to assist the fault management team in managing ambiguous information. These techniques should indicate the set of possible interpretations of the information, the activities required to reduce ambiguity, and an assessment of the impacts of performing these activities. Display of this information should dynamically change during the process of reducing ambiguity.

Fault ambiguity occurs when two or more faults are indistinguishable using the available evidence. Fault ambiguity is a common form of ambiguity in flight operations, resulting from insufficient evidence to distinguish between faults. Insufficient evidence can result from

inadequate sensors on-orbit (due to vehicle weight constraints), from limitations in the bandwidth of the downlisted telemetry stream, or when unanticipated situations occur (see also section 4.2.3).

Availability of Information

As discussed in section 3.2.3, dynamic information may be either available (i.e., the most current value is accessible) or static (i.e., not updated to current value due to inaccessible data source). Information can become static at Loss of Signal (LOS) due to the geometric configuration of the elements of the transmission system or due to a failure in the communications system. Since dynamic information that updates at a slow rate or that is currently not changing value can appear to be static, the availability of information should be indicated to agents of the fault management team. Additionally, the intelligent system should be designed to allow continued operations during periods of static information.

Problem: Managing Information Loss due to Static Data

Recommendation: The intelligent system should be designed to handle periods of static information. The user interface should display the availability of dynamic information. Since it is often important to know the last value of a parameter prior to going static, the technique for displaying static availability should not preclude displaying the last parameter value.

Example: In figure 4-38, both the intelligent system and the operator are unaware that Loss of Signal (LOS) has occurred during the MPM Deploy. They mis-interpret the lack of change in MPM state as a failure of the Deploy when it is actually due to LOS. In figure 4-39, LOS is clearly indicated on the display and data are no longer mis-interpreted. This example is based on Scenario 2, with the addition of a Loss of Signal midway through the MPM Deploy.

Techniques to indicate static data on the user interface were observed in the case study. An ascii character "s" was positioned to the right of an information item when static. Color was also used to indicate availability (e.g., use of orange to indicate static information for some of the RTDS intelligent systems).

A partial loss of information may occur instead of a total loss of dynamic information. A partial data frame may result from either reducing the number of parameters transmitted from the number in a normal data frame or by transmitting a complete set of parameters at a lower rate than usual. For example, the Space Shuttle downlists a reduced number of parameters when the high frequency transponder fails. The backup capability is a low frequency transponder, which has a significantly reduced transmission bandwidth. Procedurally, the subset of parameters that are most critical to safety and mission are downlisted. In this situation, an intelligent system would be required to function using partial information or shutdown until a full data set was acquired. One means of operating with partial data is to partition the knowledge base such that portions operating on the unavailable information can be disabled (section 4.1.3 on overriding the intelligent system). Another approach is to load a separate knowledge base designed for operation with the reduced data set. Alternately, the unavailable data could be simulated, inferred, input by the operator or ignored, depending on the desired fidelity of operation.

Problem: Managing Information Loss Due to Static Data

BEFORE - Example Illustrating Problem:

In this example, Loss of Signal (LOS) occurs during the MPM Deploy. Since there is no indication of LOS on the display or in the data set, both the operator and the intelligent system fail to notice the LOS and erroneously conclude Deploy failed based on fact that data does not change after 68 seconds.

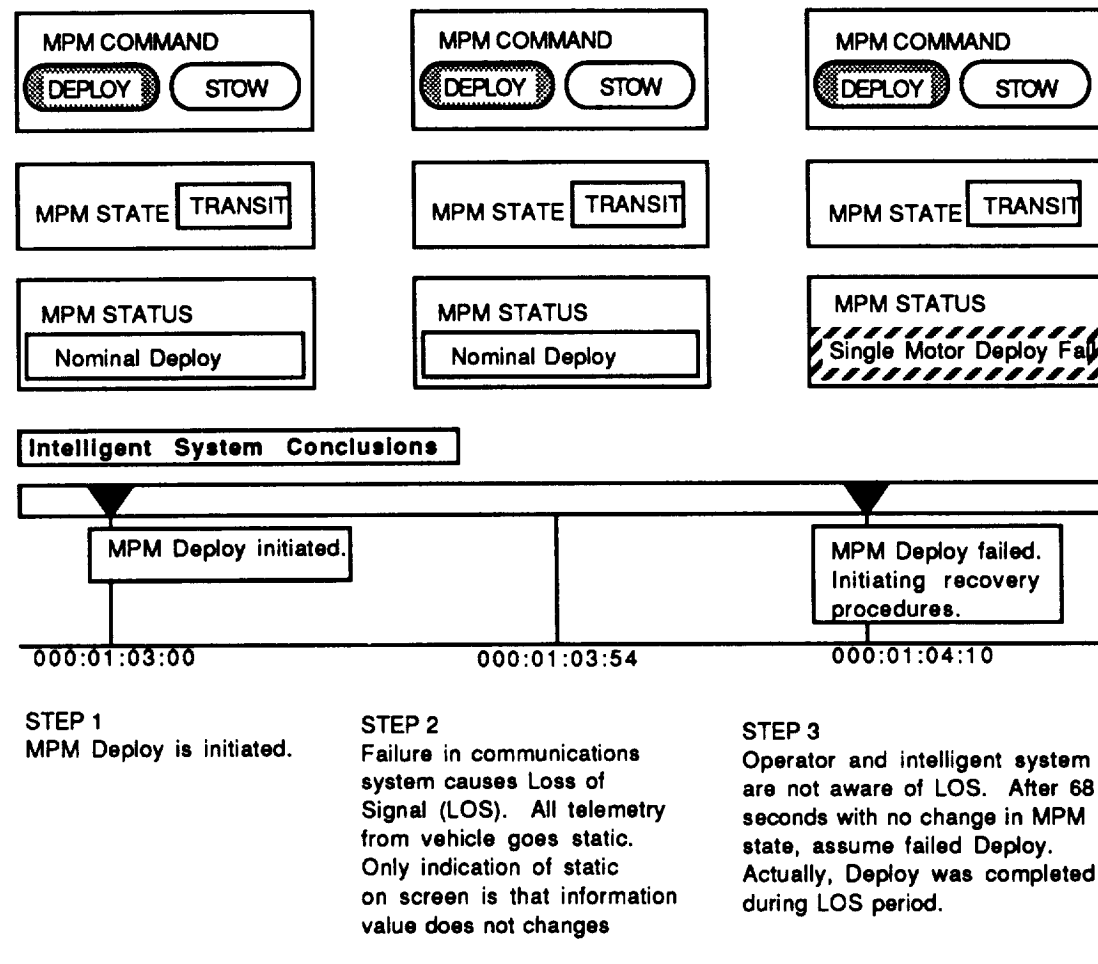


Figure 4-38. BEFORE: Example Illustrating Data Mis-interpretation when Fault Management Team is Unaware that Data are Static

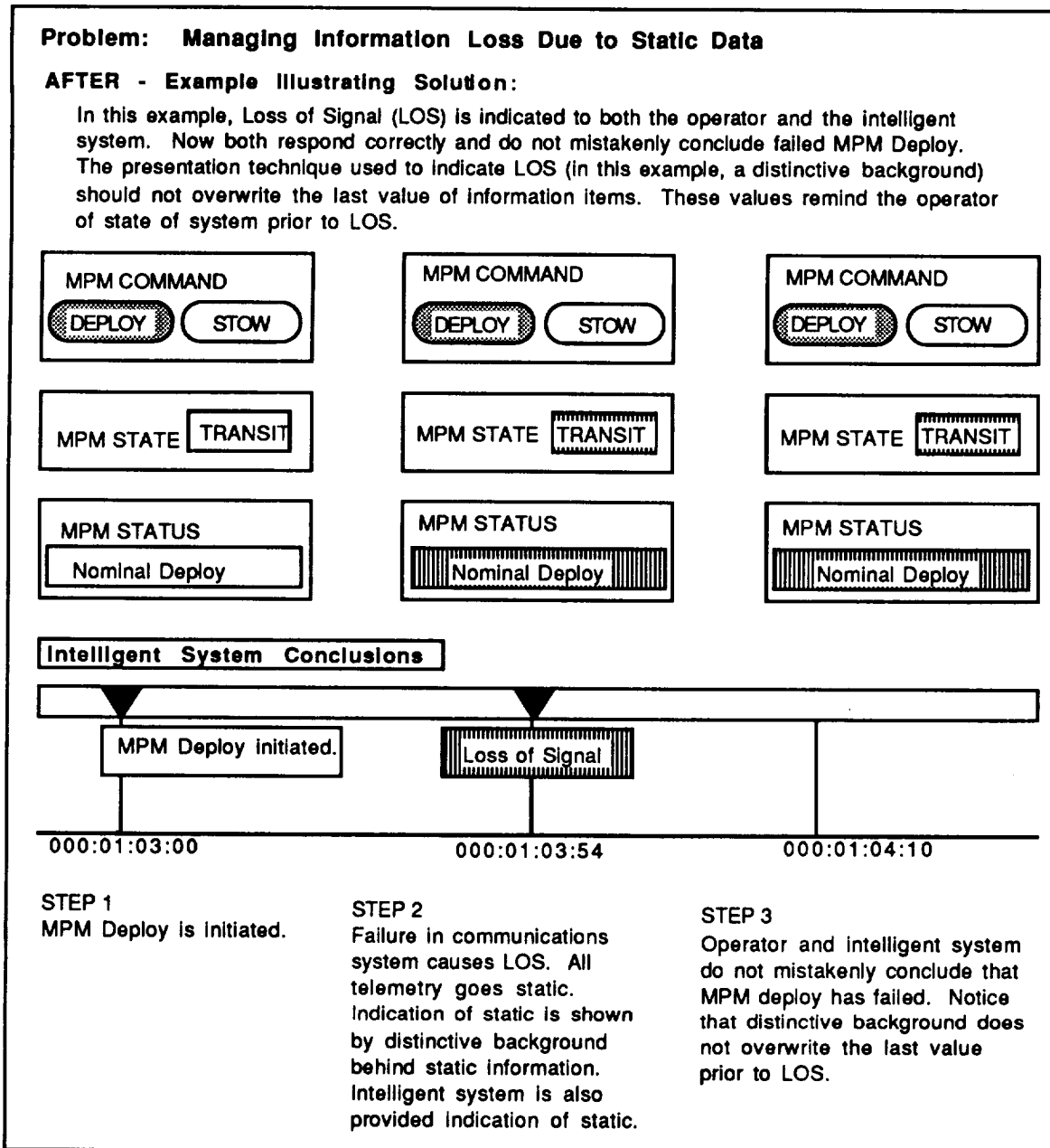


Figure 4-39. AFTER: Example Illustrating Proper Interpretation of Static Data when LOS is Clearly Indicated on Display

Problem: Operating with Partial Data Set

Recommendation: If a partial loss of data is possible, the intelligent system should be designed to recognize and operate with a reduced set of dynamic information. The desired functionality with partial data should be specified and can range from low activity, maintenance operations (e.g., record information) to active compensation for information loss (e.g., use of simulation or inference to "fill in" for unavailable information).

There are also user interface issues associated with use of a partial data set. The focus of these issues is maintaining operator awareness of important behavior in the presence of partial information (see also section 4.2.2 for a discussion of critical diagnostic information).

Problem: Operator Situational Awareness with Partial Data

Recommendation: The user interface should clearly indicate when operating with partial information. Alteration of intelligent system capability or loss of functionality during periods of partial information should be brought to the operator's attention. Any means of compensating for the loss of information should be clearly identified. The display formats provided for support during periods of partial data should emphasize maintaining awareness of critical events and anomalous behavior where possible.

Since operation with partial information can be viewed as a distinct mode of operation, the recommendations in section 4.1.1 concerning mode of operation can be useful when designing this portion of the intelligent system. If simulation or inference is to be used to estimate the unavailable information, the recommendations about display of prediction in section 4.2.2 should be considered. Note that a request for unavailable information by the intelligent system can also be a mechanism for alerting the operator about a loss of data. See the discussion of different mechanisms for user input of unavailable information earlier in this section.

4.3.1.2 Qualitative Representation

The representations of qualitative physics provide a communications medium that seems to match well with what is expected by human members of an engineering team. The ability to recognize what changes are important relies on a symbolic, qualitative understanding of the artifact. For example, an engineer who declared that a power plant was in a completely different state because the feedwater dropped from 70.0 °F to 69.00 °F would be viewed with some skepticism by his or her colleagues. Furthermore, abstracting away unimportant details is crucial for efficient communication. Qualitative representation is effective at expressing:

- **Relevance**
Qualitative criteria are commonly used to describe when models are valid and actions are applicable. Thus qualitative physics provides a crucial substrate for engineering knowledge.

- **Naturalness**
Experts use and expect qualitative language in consultations regarding monitored processes. Human engineers assume a shared background of commonsense knowledge in communications concerning artifacts.
- **Abstraction**
Qualitative representations provide a concise summary of classes of situations and behavior, making communication about complex events and states easier.
- **Minimality**
Qualitative reasoning allows conclusions to be drawn with very little information. While often weak, these conclusions are often sufficient for the task at hand, and otherwise form the basis for guiding the application of more detailed knowledge.

Problem: Information Representation for Human Communication

Recommendation: Qualitative representation can be used to organize and manage information. Since human engineers often seem to use qualitative language in discourse, many default presentations of information should be cast in qualitative terms. Such displays could be made very compact, with details presented as needed.

Qualitative representations have been developed for:

- **Numbers and equations**
The qualitative, symbolic representation of the artifact's state and behavior provides a central, integrating representation, a framework which provides the context in which the quantitative information makes sense. Typically, symbolic representation of numeric entities are organized around inequalities. For example,
 - Limit point: comparisons made with other parameters of the model that signify when important behavioral changes occur (Forbus, 1984)
 - Landmark value: comparisons between particular values taken by a parameter during a behavior (Kuipers, 1986)
 - Order of magnitude: notation of which effects dominate others (Raiman, 1986; Mavrovouniotis and Stephanopolous, 1987)
- **Ontologies**
Qualitative physics has contributed formal languages for organizing models of engineering domains. For example,
 - Device-centered ontology: concept of idealized devices of system dynamics (de Kleer and Brown, 1984; Williams, 1984)
 - Process-centered ontology: concept of physical process (Forbus, 1984)
- **Behavior**
Examples of qualitative representation of behavior are:
 - Representation of control strategy (LeClair et al., 1989)
 - Summarization of system status (Abbott, 1991)
 Behavioral representation must be supplemented by fundamental description, which is required to derive or explain the behavior. It must include changes introduced by agent actions (e.g., procedures) as well as system dynamics. Productive areas of investigation include linking behavior and function and classifying patterns of behavior, independent of role within the system.

Some examples illustrating techniques for using qualitative representation are:

- Signs of derivatives are often key indicators of system state. This information could be compactly described in a table using symbols to indicate decreasing, constant, or increasing, or perhaps using a different color or highlight pattern.
- When annotating a schematic with numerical values, it could be useful to display a parameter in terms of its quantity space. That is, set up a small scale with a discrete number of ticks and represent the value via the position of an indicator on this scale. Such displays have been very useful in (Kuipers, 1986).
- It is common to summarize unusual events in system logs. Such excursions should be noted in qualitative terms in addition to any more detailed data, to facilitate their being noticed by human team members.

Study is needed to evaluate how well qualitative representations fit with human mental models. Few such studies have been done to date (c.f. Kuipers and Kassirer, 1984).

4.3.1.3 Summarization

Visualization has been rightly hailed as a powerful advance for interacting with computers. Part of the power of visualization is summarization: We can exploit our visual perception abilities to quickly comprehend a mass of data. However, we believe visualization isn't enough. It is certainly progress to move from spending hours poring over reams of line-printer listings to watching a videotape or interactive animation. But we should be aware of the costs of visualization. It is said that a picture is worth a thousand words. But let us assume the picture is 512 X 512 pixels, with 24 bits of color, and that we can assume an average of 6 characters per word, using one byte per character. Then a picture costs 768 KB, or about 131,000 words! And while computers are getting cheaper, human beings aren't, and trained people must still watch the movies and try to understand what they mean. We suggest it would be even better for our machines to understand what we are trying to do and let them sift through the mountains of data for us. Let them report back when they have found something interesting, perhaps presenting it in video form, rather than us continuing manual labor, albeit in a slightly more pleasant form.

Issue: Information management techniques are needed that go beyond visualization, which can be costly in processing time and which still require humans to interpret the visualization. Task-dependent algorithms for summarization and suppression of irrelevant detail seem a promising extension.

This insight is based on our experience with research interfaces for qualitative model development. Correctly managing the amount of detail presented is a key issue, but we don't have very good solutions for it yet. And maybe there aren't any. No graphical interface is going to make a 10,000 state envisionment instantly comprehensible. If, on the other hand, all you care about are steady-state situations and there are only a handful of them, a good interface can make it easy to analyze that handful of states to select the one you are interested in. Task-dependent algorithms for summarization and suppression of irrelevant detail seem a promising alternative.

4.3.2 Workspace Design

Design of the workspace includes specification of the visual appearance of the user interface and the style of interacting with it. This specification should be derived from the information requirements and agent activity description for the task and should be evaluated for utility using

operational scenarios. Designing the workspace to assist the operator when navigating through the workspace is discussed. The trade-off between complexity in the workspace design and intelligent system performance is identified and methods to compensate for these performance impacts are described. The effect of the user's level of expertise on both the visual appearance and the style of interaction with the user interface is discussed. Finally, alternatives to the most common visual form used in designing the intelligent system workspace, the message list, are investigated.

Recommendations and issues affecting both the visual appearance of the user interface and the style of interacting with it are summarized in this section. An in-depth discussion of designing for visualization of the monitored process and intelligent system activity is provided in section 5.

Navigation through Workspace

Navigation is the means of traversing the workspace to access and manipulate information. There are a number of design considerations that affect workspace navigation. The grouping of information into multiple windows that can be selectively closed to reduce screen clutter can introduce complicated and confusing interaction sequences. The operator can become lost when navigating through such a complex workspace to access required information. The operator can also become disoriented when navigating through a *process view* (i.e., a coherent unit of representation of a portions of the underlying process or systems which the observer could select for display, such as a schematic; see section 5.1.2) that is larger than the available real estate for displaying them (i.e., where only part of a view can be displayed at one time).

Thus, an important HCI design trade-off affecting workspace navigation is information density versus interaction complexity. Increasing the availability of information often complicates interaction with the workspace. A good design represents a balance between these two approaches. Approaches for achieving this balance include assisting the operator in navigating through a complex workspace or providing mechanisms for reducing information density. For complex interaction, a "map" can be provided to identify where the operator is currently located within the interaction sequence or display unit. Information density can be reduced locally by grouping non-critical information into "overlays" that can be de-selected by the operator (Chu, 1990). See also section 4.1.1 for a discussion of interaction options that vary by mode of operation.

The operator should be able to easily identify the available interaction options, both for access to and control of the monitored process and the intelligent system as well as control of the format and content of the user interface (Johns, 1990).

Problem: Maintaining Orientation in Complex Workspace

Recommendation: The workspace design should include aids for navigating through the workspace and should provide the capability to control information density on the user interface.

Example: Figures 4-40 through 4-45 illustrate techniques that assist the operator in maintaining orientation within a complex workspace. These examples are described below in more detail.

Although there are no definitive guidelines on how to make this trade-off, some suggestions for achieving balance between information density and interaction complexity are made below:

- **Navigating through Available Windows**

A graphical representation of available screen formats can be used as a map for orienting the operator and as a mechanism for moving through the workspace. See figures 4-40 and 4-41 for an illustration of presentation of information about what windows are available for display and how to improve navigation through them.

- **Navigating through Schematics and Graphics Larger than Display Region**

When navigating through a graphic larger than the display area, a representation of the entire graphic (e.g., schematic) with the visible portion highlighted within that representation is a useful orienting mechanism (Shaw, 1988). See figures 4-42 and 4-43 for an illustration of how this overview graphic assists the operator in avoiding getting "lost" in a large schematic. Also, see Johns (1990) for additional information on the navigation of schematics.

- **Selectively Hiding Unnecessary Detail**

Organizing task-specific information into operator-selectable overlays can be effective in reducing visual complexity when the information is not required for the current task (Chu, 1990). See figures 4-44 and 4-45 for an illustration of the use of overlays to reduce information density on the screen.

- **Using Fixed Regions for Quick Scanning**

A combination of fixed and dynamic regions on the workspace can assist in balancing information density and interaction complexity. A quick scan of the fixed regions can direct the operator toward a system requiring detailed investigation (i.e., what to call up in the dynamic region of a workspace).

The examples referred above are not based on one of the scenarios provided at the beginning of section 4. Adequate background to understand the example is provided in each figure.

Issue: Research is needed into methods assisting the designer in making the trade-off between information density and interaction complexity of the HCI.

A variety of techniques for workspace navigation were observed in the case study. Many systems had a workspace designed around pre-defined screens. Navigation through the workspace consisted of displaying the desired screen. Within a screen, there were often combinations of fixed display regions and regions with operator-selectable displays. Hypermedia connections between information displays were demonstrated in the GNC Air Data System.

Problem: Maintaining Orientation about Available Windows

BEFORE - Example Illustrating Problem:

In this example, it is not clear what windows are available for display or how to access available windows. The operator can become "lost" in a complex interaction sequence just trying to view another window.

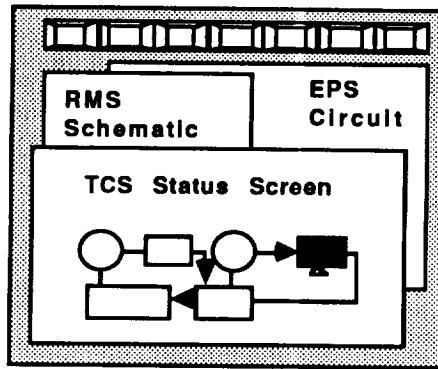


Figure 4-40. BEFORE: Example Illustrating Difficulty in Navigating Among Multiple Windows

Problem: Maintaining Orientation about Available Windows

AFTER - Example Illustrating Solution:

In this example, an overview of all available windows is always displayed at the top of the screen. The selected window is highlighted using grey tone. The operator can now easily identify what windows can be viewed and how to access them.

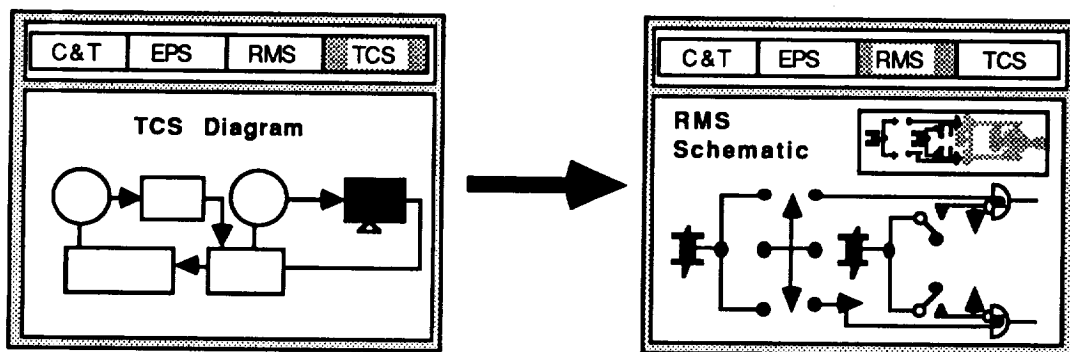


Figure 4-41. AFTER: Example Illustrating Presentation of Information about What Windows are Available and How to Navigate through Them

Problem: Maintaining Orientation In a Large Schematic

BEFORE - Example Illustrating Problem:

In this example, it is not clear if the entire schematic is visible or if only a portion of the schematic is visible. Additionally, if only a portion is visible, it is not clear which portion is visible. With such a display, the operator can easily become "lost" in the schematic.

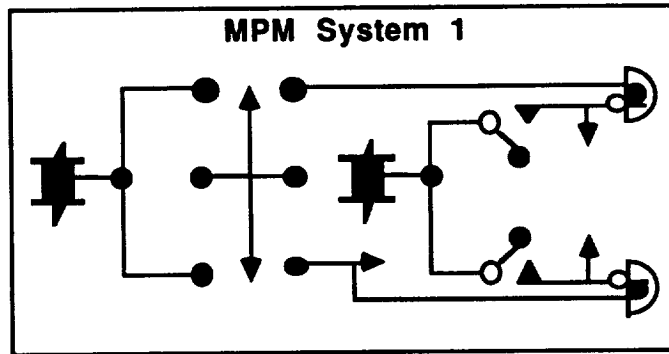


Figure 4-42. BEFORE: Example Illustrating Operator Getting "Lost" in a Schematic

Problem: Maintaining Orientation In a Large Schematic

AFTER - Example Illustrating Solution:

In this example, an overview of the entire schematic is reproduced in the upper right hand corner. The visible portion of the schematic is displayed in black and the hidden portion of the schematic is displayed in grey tone. The operator can now easily identify what portion of the schematic is visible

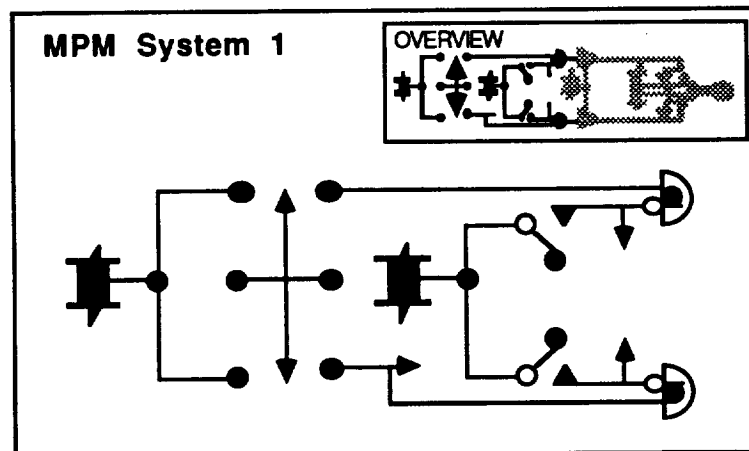


Figure 4-43. AFTER: Example Illustrating Navigation Aid for Moving Through a Schematic

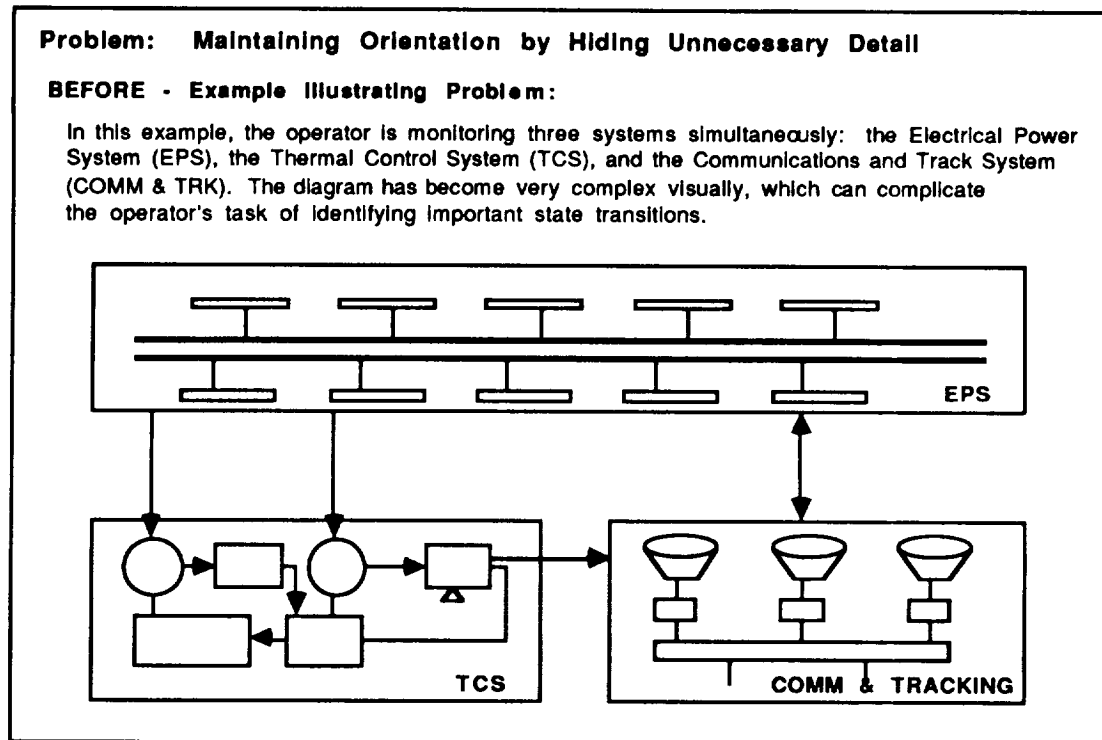


Figure 4-44. BEFORE: Example Illustrating Visual Complexity of a Display

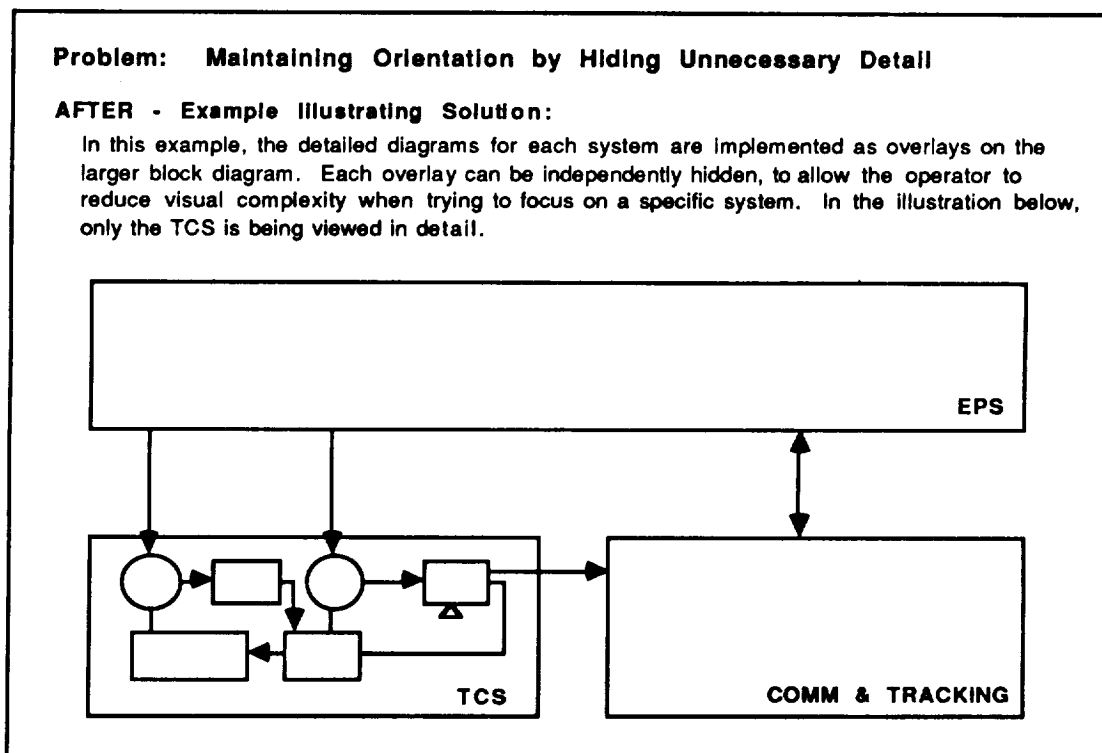


Figure 4-45. AFTER: Example Illustrating Reduction of Visual Complexity using Overlays to Hide Unnecessary Detail

The ability to balance information density and interaction complexity is complicated by the presence of more than one physical screen. The amount of visible information is usually increased by increasing the real estate. And interaction gains another dimension -- viewing and navigating through multiple screens. Issues associated with using more than one physical screen include the ability to coordinate the access of information on multiple views and to effectively move among these views. Ease of access is related to how well information is visually organized and integrated with the expected types of interaction. It is important to specify the activities to be performed so that information can be partitioned effectively between screens. With more information visible, it becomes especially important to clearly identify the important behavioral changes as they occur (e.g., violation of critical boundaries, major events). The ability to quickly scan for anomalies is necessary in this type of support environment (section 4.2.2).

Issue: Investigation is needed into display and navigation issues when more than one screen is visible at a time. This investigation should include issues associated with integrating intelligent system technology into an existing operational environment.

Ground operations for both the Space Shuttle and the Space Station are likely to include user interfaces that span more than one physical screen. For Space Shuttle, the recent upgrade to workstations for flight support has not precluded the use of the displays on the existing flight support consoles. In this situation, HCI with the workstation should be integrated with use of the original displays. See section 6.3 for more information on integration of new technology into an existing operational environment.

Design for Performance

Design is a process of performing trade-offs between conflicting alternatives (Norman, 1983). In the previous section, the trade-off between information density and interaction complexity was discussed. Another important trade-off for HCI design is the availability of information versus the performance of the intelligent system. During the case study, intelligent system performance problems were observed in some systems with complex HCI capability implemented using advanced software (e.g., G2[®], The X Window System[™]). The workspace should be designed to balance the availability of information (i.e., complexity of interface) with the intelligent system performance impacts of making that information available. This trade-off could include evaluation of a prototype for performance impacts. As a part of such an evaluation, the ability of the prototyped design to scale up to a full capability HCI should be assessed.

Issue: Research is needed into methods assisting the designer in making the trade-off between availability of information and the performance of the intelligent system.

In a real-time support environment, performance is an important measure of the utility of support software. If the software cannot provide timely results, it is of limited use. A flexible HCI design should include provision to accommodate potential performance problems. Essential capability for mission support should be distinguished from desired capability. The system should be designed to allow the operator some control over system performance (e.g., disabling non-essential capability as needed to improve performance). Techniques for improving performance include off-loading segments of the knowledge base, reducing rate of information input to the intelligent system, or disabling portions (elements) of system

[®] G2 is a registered trademark of Gensym Corporation.

[™] The X Window System is a trademark of MIT.

functionality. Figure 4-29 in section 4.1.3 illustrates the technique of disabling portions of the knowledge base.

User Expertise

Methods of interacting with the user interface that are obvious to the novice user may be cumbersome or disruptive to the expert user (Norman, 1983). For example, the use of a long series of informative menus to access an information item may provide necessary training for the novice about available display options. Once the novice is trained, however, these menus lose their utility for that user and can become an obstacle in performing operational tasks. Means of bypassing these orienting mechanisms should be provided. The content and presentation of the user interface should be tailored to the expertise level of the operator (Wick, 1989).

Problem: Supporting Both Novice and Experienced Users

Recommendation: If the operational user interface is to be used for training as well as operational support, the use of redundant types of interaction and alternate forms of presentation can accommodate the conflicting needs of the novice and the expert while retaining the same functionality (Norman, 1983). Display formats and interaction methods can be varied to tailor the content and presentation of the HCI to the expertise level of the operator (Wick, 1989).

Example: Figure 4-46 illustrates a complex interaction sequence arising from nested menus. Figure 4-47 shows a redundant, alternative method for interacting that reduces complexity. This example is derived from the RTDS applications in the case study. See the discussion below for additional information about this case.

An example of this concept was observed in some of the RTDS applications (Volume 2, Malin et al., 1991). The user interface provided mouse-selectable, pull-down menus for interaction with the system. These menus provide a good orienting mechanism for the novice operator by displaying available options at each menu level and by leaving a visual trail of the inputs required to perform an action. In some cases, however, these menus were nested deeply and interaction became cumbersome to experienced operators. An alternate, redundant method of interaction was provided in the form of function keys on the keyboard. This allowed experienced operators to quickly access desired functionality without proceeding through the menu hierarchy (Gnabasik, 1990). Figures 4-46 and 4-47 illustrate this example.

It is important for the system to use the same terminology as used by experienced operators. When an intelligent system is to be used by inexperienced operators, this reinforces other forms of training through use of a common language. It is imperative that the intelligent system use the "operational" language with experienced operators as well. An "operational" language typically exists to prevent ambiguous statements in critical situations. The intelligent system must use that language to be clearly understood and to be accepted by experienced operators.

Problem: Supporting Both Novice and Experienced Users

BEFORE - Example Illustrating Problem:

In this example, the operator must use the menus to interact with the intelligent system. For complex systems, there can many, nested menus. Multiple mouse-clicks may be required for one operator input. Although the menu system is very useful when learning how to use the user interface, it can be frustrating to an experienced user.

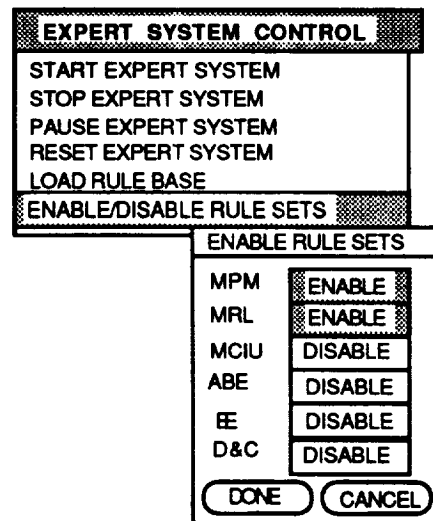


Figure 4-46. BEFORE: Example Illustrating Complex Interaction Sequence using Menus

Problem: Supporting Both Novice and Experienced Users

AFTER - Example Illustrating Solution:

In this example, simple keystrokes (e.g., control E) can be used to bypass the menus and go directly to a window. Here, the operator enters " ^ E " and the window "ENABLE RULE SETS" is immediately displayed.

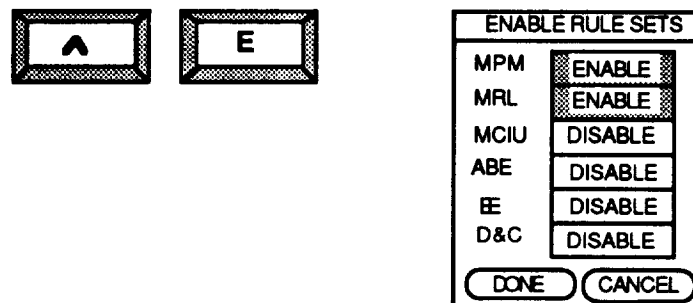


Figure 4-47. AFTER: Example Illustrating Alternate Method of Interaction using Control Keys

Problem: Avoiding Unfamiliar Terminology

Recommendation: Use the terminology of the operators in the user interface. When well-accepted terminology does not exist for a concept, clearly define all new terminology and consult experienced operators in the selection of that terminology.

Types of User Interfaces

All intelligent systems are operated by at least two types of users: personnel who make operational use of the intelligent system and personnel who develop and maintain the intelligent system. Not all systems distinguish between these types, however. Often, the user interface built for system development becomes the user interface for operational support without any recognition of the unique needs of these different users. In many of the cases surveyed, however, the concept of a user interface that differentiated between development and operations use had been considered. Typically, the development user interface was a super-set of the operational user interface, providing access to extra information or additional types of control that are not needed during operational use.

Problem: Providing an Operational User Interface Distinct from the System Developer Interface

Recommendation: Operational users have information needs that differ from those of system developers and maintainers. The operational user interface should be distinct from the system development user interface and based on operational information requirements.

For model-based intelligent systems, there are three distinct classes of interface users, each with somewhat different, although overlapping, requirements. *Domain Model Developers* are involved in creating generic domain models that can be applied to a wide variety of specific physical systems. These generic domain models constitute "off the shelf" knowledge bases that can be used for developing models of a wide variety of monitored processes. In particular, the next class of interface users, *System Model Developers*, use these domain models to construct models of specific physical systems for particular tasks. *Model Users* interact with the finished intelligent system. As qualitative modeling technology becomes more widely used, it is less likely that the same people will be involved in all three roles.

All three classes of users need better interfaces than are available today. Domain Model Developers need the ability to visualize large, complex envisionments, to more quickly ascertain the consequences of changes in a model. Sophisticated model analysis tools, which could for example compare envisionments produced by different domain models across a suite of test cases, could substantially speed up the development of "off the shelf" knowledge bases. System Model Developers need the ability to interactively select the appropriate domain constructs and levels of detail for their particular task. An intelligent system that collaboratively helped the System Model Developer identify implicit modeling assumptions and navigate through a space of models varying both in grain size and technology could speed the development of qualitative models for specific applications.

Model Users at minimum need the ability to extract enough information from the results generated using the qualitative model to have faith in its conclusions. Model Users engaged in sophisticated problem-solving will also need many of the tools needed by System Model Developers, since they will be reasoning about fault hypotheses, many of which can be represented as alternate modeling assumptions. Model Users will also need interfaces which aid the "tracking" of what is happening in the system with model predictions. For example, consider a fault-management scenario. Suppose the user, in collaboration with the intelligent assistant, has narrowed a problem down to one of two failures. The interface must make clear (a) what consequences the competing hypotheses have in common and (b) what consequences of them differ. The interface should simplify the identification of predictions which can distinguish between the hypotheses. Alternately, consider a monitoring scenario. The user, knowing of a future thermal demand that could overload the thermal control system, might desire a "watch routine" that would look for signs that the system is overloading or confirm that an overload will not occur. By assuming a particular future state, the assistant could generate predictions concerning measurable parameters, including criteria for invalidating the hypothesis of overload (such as coolant temperature leveling off at a higher value after the load, but still cool enough to allow significant heat transfer). The interface must make clear (a) what the nature of the watch's conditions are, (b) be able to explain why those are appropriate, and (c) accurately inform the user when the set conditions either come to pass or fail to do so.

Lists and Schematics

The message list and the schematic are the two most common display forms used in the development of intelligent systems. The designer should be cautious about using such display forms simply because they are commonly used and focus instead on using a representation that is most effective at communicating the desired information.

Most systems surveyed in the case study used a message list to display messages from the intelligent system (Volume 2, Malin et al., 1991). The message list consists of time-sorted message fields defining important events, anomalous behavior, or intelligent system assessments and recommendations. Often, a scroll bar provides access to messages describing behavior in the past. Figure 4-48 shows an example of a message list.

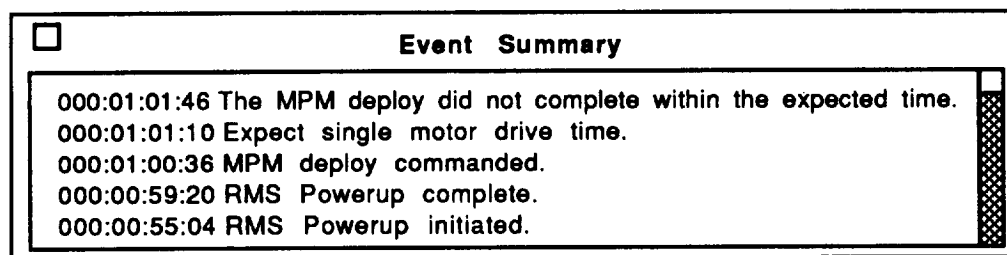


Figure 4-48. Example of Message List

The message list has some drawbacks as a representational form, however. It does not reveal causal or structural relationships between messages (e.g., associate messages from a given hardware system). If specific timetags are not included, it often does not reveal temporal relationships either. It can be difficult to emphasize important information from less important information. Alternatives to the message list include a timeline representation (see PDRS Decision Support System in Volume 2, Malin et al., 1991), plots of numeric information, or animated graphics, such as schematics.

Issue: Effective alternatives to the message list as a format for display of events and recommendations are needed.

The most common use for schematics was to represent the physical topology of the monitored process. Often these representations are based on existing paper-based diagrams. There is a tendency to reproduce such schematics because they provide a compact domain representation. Such forms are only useful, however, if they assist in interpreting the information to be presented. For example, if the location of components is important in interpreting information, then a physical topology diagram is appropriate. If the information concerns state, however, a functional representation would be more effective. See section 5 for detailed discussion and recommendations about both message lists and schematics.

Section 5 Design for Visualization of the Monitored Process and Intelligent System Activity

5.1 Introduction

Some of the trends that we will discuss relate to the representational windows available to help the human operator visualize the state of the monitored process in relation to the intelligent system's assessment and actions. What kinds of windows did the intelligent system designer make available to the human operator? Do they help the operator see through the interaction barriers shown in figures 1-2 through 1-4?

Ironically, the net result of the application computer interface technology (e.g., windows and graphics) in some of the applications is to increase the demands on slow, deliberative, capacity-limited user cognitive processes rather than engage parallel, automatic, perceptual recognition based cognitive processes. Examples of this include:

- Navigation and interface control mechanisms that place high demands on user memory rather than use the interface as an external memory
- Navigation demands that shift the user's focus of attention away from the monitored process and towards the interface itself
- Display forms that force a serial process of finding, collecting and integrating individual data elements (e.g., message lists) rather than organizing related data to make critical states and events "pop out."

The irony is heightened by the fact that, in contrast to the above, there were several cases that supported better visualization of the monitored process (e.g., ILDSS), better information handling capabilities that reduced data overload problems (e.g., Selective Monitoring System, SELMON), and new conceptualization aids that helped the user search for and discover patterns in the data (e.g., some features in SHARP).

In the following sections, we examine several trends observed in the case study -- workspace, schematics, and message lists -- to point out problems and possible new directions. Again, this is work in progress so that these sections are samples of issues about how to support effective human-intelligent system cooperation. To provide some structure about the multidimensional nature of the interface capabilities that support human interaction, we first provide an overview of the design for information transfer criterion and different levels of description for computer based display systems.

5.1.1 Design for Information Transfer Philosophy

After Gibson (1979, p. 42), one can define a display as, a surface shaped or processed so as to exhibit information about more than just the surface itself. Given this definition¹, the design of displays/interfaces/aids is shaping or processing display surfaces (the medium, e.g., video display units (VDUs), and the elemental domain data) so as to exhibit information for a domain practitioner (the problem solver who is also the problem holder). One can term this approach design for information extraction (Woods, 1991). This criterion emphasizes that effective

¹ And given a specific view of information where information is not a thing-in-itself, but rather a relation between the data, the world the data refers to, and the observer's expectations, intentions, and interests. Informativeness is not a property of the data field alone, but is a relation between the observer and the data field (cf., Woods, 1986).

interfaces/displays must do more than provide access to domain data; they must assist the practitioner extracting the information needed to perform better. The displays/interface should support the operator in seeing, assessing and acting on the underlying process through the computer medium. Thus, the computer displays/interface form a representation of the underlying process, i.e., the monitored process, the intelligent system, and their interaction (figure 1-4).

Representational form is defined in terms of how data on the state and semantics of the domain is MAPPED into the syntax and dynamics of visual forms in order to produce information transfer to the agent using the representation. It is indifferent to the visual form per se. Bar charts, trends and other visual forms can be used to create the same representational form (or variants on a representational theme). Conversely, the visual appearance alone does not define the representational form so that a single visual format such as a bar chart can be used in many representational forms.

The design challenge for creating effective representations is to set up this mapping so that the observer can extract information about task-meaningful semantics. Becker and Cleveland (1984) call this the decoding problem, that is, domain data may be cleverly encoded into the attributes of a visual form, but, unless observers can effectively decode the representation to extract relevant information, the representation will fail to support the practitioner. This mapping challenge is central to the discussion of problematic trends and possible new directions in the following sections.

5.1.2 Levels of Description of a Computer-Based Display System

To describe the trends and problems, we first need to introduce the concept that there are several levels of analysis at which a designer can think about a computer based display system (Woods and Eastman, 1989; Woods, in preparation). We use these different levels of analysis as a structuring heuristic for organizing design techniques and practices for creating visualizations of dynamic processes (figure 5-1). Many of the terms that we will introduce here will be used to describe trends that we observed in interface capabilities designed to support human interaction with intelligent systems.

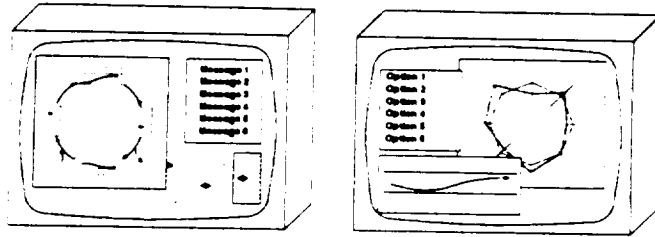
The graphic form level is the fundamental hinge in the series of levels. Design activities below that level are concerned with crafting graphic forms out of elements (for example, reference values, numeric scales, unitizers, labels) and atoms (e.g., lines, characters). Above the level of forms, design activities are concerned with grouping, organizing and coordinating forms to create process views and a workspace.

At the highest level design focuses on **WORKSPACE COORDINATION**: coordinating the set of viewports and classes of process views that can be seen together in parallel or in series.

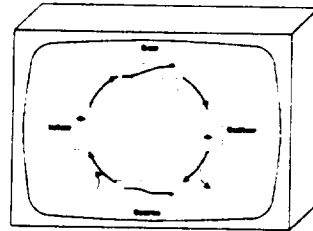
Viewport: Any screen real estate that serves as a unit where process views can appear. A viewport can be a region within a single VDU -- a window or a whole VDU screen. A set of viewports, i.e., a workspace, can consist of multiple window viewports, multiple VDU viewports and any combination of the two.

Process View: A coherent unit of representation of a portion of the underlying process or systems which the observer could select for display in a viewport.

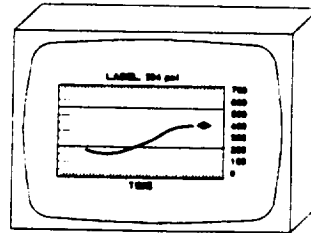
Workspace



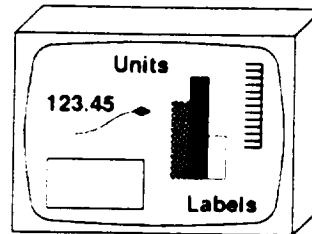
Process Views



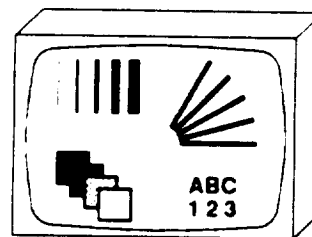
Formats



Elements



Atom



© 1991 Woods, Potter, Johannessen, Holloway

Figure 5-1. Levels of Analysis in Computer-Based Display Systems

Workspace: Set of viewports and classes of process views (content) that can be seen in parallel and in sequence as a function of context. In the extreme only one viewport is available and each process view takes up the entire viewport (the historical default of a "display page". The workspace can include multiple windows and/or multiple VDUs as viewports.

The workspace includes the classes of process views that are available and their inter-relationships. Design of the workspace requires specification of how the classes of process views are mapped into the available viewports, i.e., a set of coordinated viewports/display classes. Total flexibility, i.e., any process view can appear in any viewport as the observer chooses, represents a failure to design the workspace (e.g., Moray, 1986; Henderson and Card, 1987; Cook et al., 1990).

One of the critical forcing functions in the design of data displays for the computer medium is that the set of potentially observable display units or chunks is very much larger than the available viewports (physical display area or real estate). This characteristic of computer based display systems creates the danger of the keyhole effect where the user is unable to maintain a broad overview, becomes disoriented, fixated, or "lost" in the display structure (Woods, 1984; Elm and Woods, 1985).

At the workspace level, there are several issues of concern. These are:

- How is the information coordinated
- How does the user know where to look next?
- Can the person find the right data at the right time?

Some of the trends in the case study at this level of analysis are:

- Proliferation of windows, typically each specialized for just one type of data
- Navigation issues on where to find related data, especially complicated by hidden windows and complex menus
- Excessive flexibility in user tailored environments (e.g., user controllable window configuration) that creates data management burdens

At the level of process views, the designer is concerned with orchestrating a set of graphic forms to form a larger representational tapestry with respect to a new level of domain issues. A process view could consist of one graphic form or a group of graphic forms (e.g., schematics, timeline displays). The historical default case is where each process view takes up the entire viewport (and a viewport consists of a whole VDU screen) -- a "display page". Note that with "automatic" display creation via intelligent processing process views may not necessarily be pre-designed but rather created or assembled or modified adaptively or "on-the-fly."

The critical design challenge here is to break the process into a set of coherent views or kinds of views -- defining the boundaries for collecting together sets of domain data. The boundaries may be drawn based on process system lines, process functions, control systems, operator tasks, or some combination.

The major trend in the case study at this level of analysis is:

- Inscrutable systems where the dominant process views available, physical topology schematics of the monitored process and message lists as output from the intelligent system, do not help the operator see patterns of events

The graphic forms level deals with issues in designing representations of some part or function of the target world. The basic design focus at this level is -- what data sets and data relationships need to be represented to transfer information to the observer (i.e., the mapping question)? The main design activity is crafting integrated representational forms from elements and other forms. There are two design goals for the design of representational form: (1) how does the form highlight and support recognition of anomalies? (2) how does the form reveal the dynamics of the process and highlight changes (e.g., behavior over time, events, movement towards or away from landmark states, what has happened, what will happen next).

The major trend in the case study at this level of analysis is:

- Over-reliance on digital/linguistic forms of representation resulting in the dissociation of related data, failure to highlight anomalies, failure to highlight state changes and dynamic behaviors

Elements are the building blocks manipulated to craft graphic forms and consist of items such as reference values, numeric scales, unitizers, labels, indicators, digital values, linguistic strings, icons.

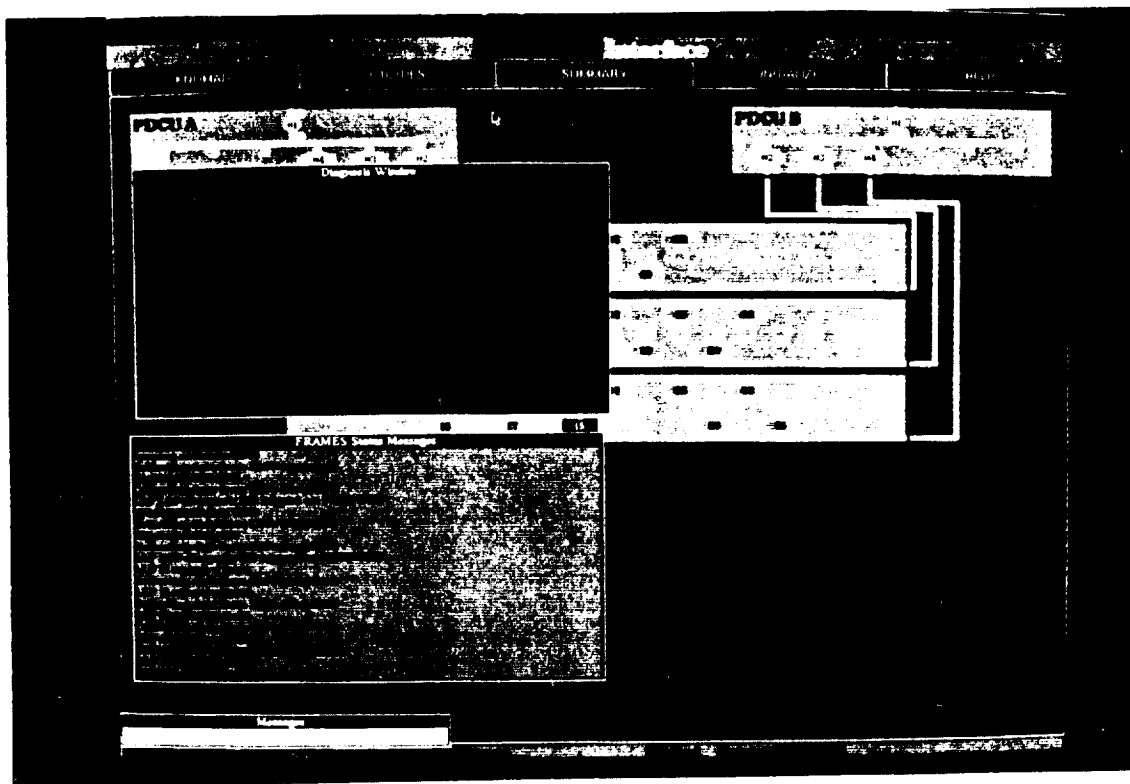
Elements are in turn made up of what we call the "atoms" of display design. The atoms generally are concerned with limiting factors -- pixel resolution, character sizes, fonts, aspect ratios, line widths, color/intensity palette.

If the collection of display frames in a system could be conceived of as one huge layout, the levels may be thought of as a series of nested spaces, from the overall workspace design, to the forms that occur within process views and viewports, to the elements and atoms used to craft the forms. The designer zooms back and forth between these nested spaces during the design process (cf., figure 5-1).

5.2 Workspace - Proliferation of Windows

The use of workstations with high-resolution monitors and window-oriented development shells creates a computer workspace where the user can call up different kinds of process views into the different available viewports (windows) on one or more VDUs. Design of the workspace requires specification of how the classes of process views are mapped into the available viewports, i.e., a set of coordinated viewports/display classes. A critical forcing function on the design of the workspace is the fact that the set of potentially observable displays is very much larger than the viewport space available in parallel (physical display real estate).

Figure 5-2 shows a common theme observed in the case study. There is a base set of viewports visible on each VDU with a default set of displays and interface control mechanisms (soft function keys, pull down menus) presented in these windows. There may be other process views which can be called into the base viewports displacing the default displays. In addition, the user can open additional windows on top of the base viewports, partially or totally obscuring the view displayed within the base viewport. In the example in figure 5-2, a physical topology schematic is the default process view displayed within a base viewport. The user has opened two additional viewports which contain diagnostic and status message lists on top of this base.



There is a base set of viewports visible on each VDU with a default set of displays and interface control mechanisms presented in these windows. In addition, the user can open additional windows on top of the base viewports, partially or totally obscuring the view displayed within the base viewport. In this example from the case study a physical topology schematic is the default process view displayed within a base viewport. The user has opened two additional viewports which contain diagnostic and status message lists on top of this base.

Figure 5-2. Common Theme in Workspace Design: Default Workspace Augmented with User Configurability

This approach provides the user with a default workspace plus user configurability. The user has great flexibility in reconfiguring the placement and size of windows. The user can call up and configure many additional viewports/process views. The user can configure the viewports so as to be able to see the contents of two viewports in parallel.

However, too great a reliance on user configurability represents a failure to design the workspace -- the burden of determining what sets of process views should be seen together and the data management activity to configure this collection of views should not fall exclusively on the operator of dynamic systems during real-time operations. This is one way to produce clumsy automation where the user's data management burden is high during high tempo, high workload operating conditions such as fault management (cf., Woods, 1984; Woods and Elm, 1985; Henderson and Card, 1987; Woods et al., 1990; Cook et al., 1990 for research on the effects of workspace design and on design techniques and principles to create more effective computer workspaces).

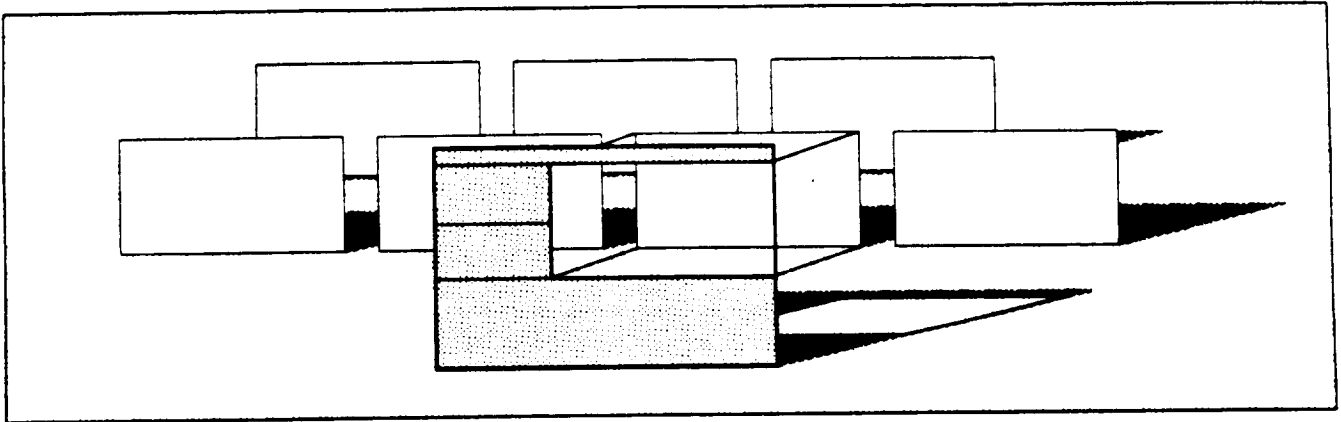
Excessive user configurability creates extra data management tasks that shift operator attention to the interface (where is the desired data located in the display space?) and to interface control (how to I navigate to that location in the display space?) at times where his or her attention needs to be devoted to assessing/managing the monitored process.

In general in the case study, the interface control mechanisms used to navigate within the display space tend to divert operator attention from the monitored process to the interface itself. Interface control mechanisms frequently imposed extra information processing steps and user memory burdens due to hidden windows, hidden menus, and unnecessary dialog steps (the interface was underutilized as an external memory).

This lack of transparency is particularly problematic because of another trend. There was a tendency for windows to proliferate, each specialized for just one type of data. This fragmentation or dissociation of data across different windows forces the operator into a slow serial search to collect and then integrate related data. The proliferation of windows degrades rather than supports the cognitive component of interface navigation -- knowing where to look next and finding the right data at the right time (Woods, 1984; Elm and Woods, 1985).

Figure 5-3 illustrates another common feature of the workspaces surveyed in which several different views can be selected for display in a given viewport. In this approach (commonly referred to as tiled windows), there are no overlapping process views so that the operator is not moving windows out of the way to see what's happening on the schematic display. However, this approach involves other important issues that relate to serial versus parallel display of data, how does the operator know when he should examine a different view, how is an event on a non-displayed view signaled to the operator.

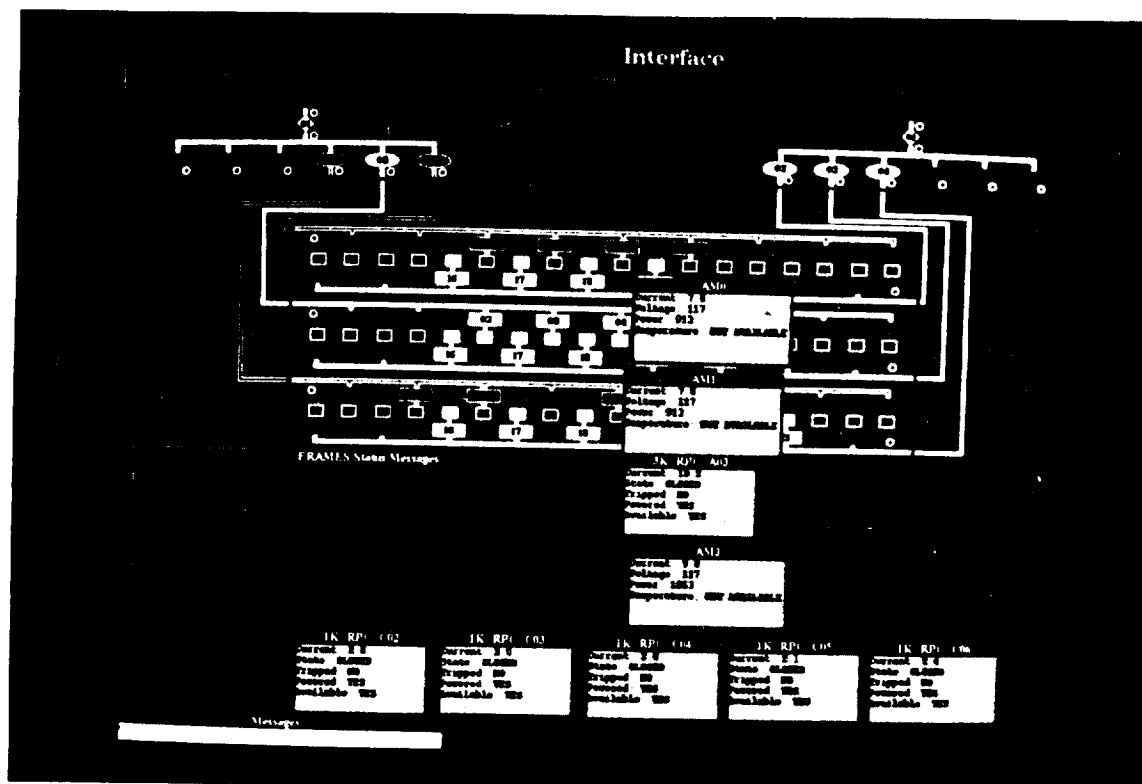
An important issue in workspace design is how to coordinate different views so that the operator can easily find relevant information. The proliferation of windows can create a situation where many viewports are opened on the VDU at one time. Figure 5-4 illustrates how the VDU can be cluttered with windows causing problems in determining where to focus attention and making it easy for the operator to miss new events. In addition, note how this creates a new operator data management task -- de-cluttering the workspace, where the user must remember to remove stale views/viewports. If the user does not de-clutter early, he or she may miss significant changes or find that they need to do the de-cluttering task at the same time that they should be assessing the effects of a changed process state or evaluating how to respond to the change. Forcing the operator to work with only relatively few tiled windows does not solve the overall problem (although it can reduce de-cluttering demands) since the operator may miss significant events on non-displayed views or need to go through a serial process to find and compare related data that appears on different views.



© 1991 Woods, Potter, Johannesen, Holloway

In this example, there are seven different views which can be examined serially in the main window (upper right).

Figure 5-3. Second Approach to Workspace Design: Tiled Windows With User Selecting Which View is Presented



ORIGINAL PAGE IS
OF POOR QUALITY

Figure 5-4. Example of Proliferation of Windows and Resulting Clutter Which Impairs Locating Important Information

One of the key questions to be addressed at the workspace level is what information needs to be viewed at the same time and what can be presented serially. The answer to this question comes from a cognitive task analysis -- what information should the observer be able to extract from observing this display (in isolation and in concert with other displays)? This question addresses the issue of what role a particular display (window) plays in conjunction with other information available. Figure 5-5 presents one approach to this problem by including a "scratchpad" window onto which information from other windows can be copied. In this manner, the user can select what information to view in parallel.

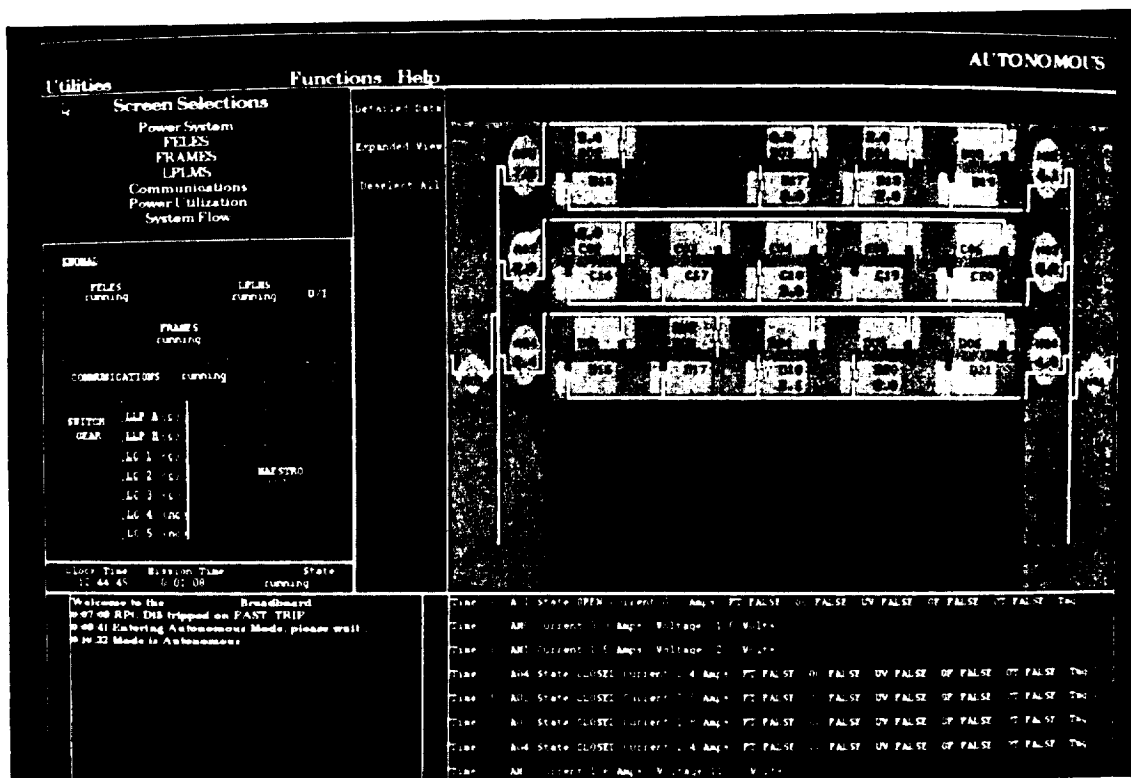
Reminding can take the form of informing the user about planned actions, past activity, state of the system (e.g., mode), etc. The key aspect of this principle is to relieve excessive memory burdens on the user. According to Norman (1988), the ideal reminder must contain two components, the signal (that something is to be remembered) and the message (what is to be remembered). With overlapping windows, the potential for obscuring reminders exists. As figure 5-6 indicates, overlapping windows violates this principle. In this example, the window which was called up covers relevant data which needs to be compared to the information in the new window. In general, this is a problem of having background, hidden windows. Another example violating this principle was presented in figure 5-2.

In one system we saw, after the intelligent system made some diagnosis, a message informing the operator of this fact would appear. However, if the operator wanted to see the diagnosis, he would have to select the menu and menu item that brought up this window. This example shows how there is often a failure to use the computer power ready at hand to assist the operator's in handling the large amounts of available data -- the data overload problem. If the intelligent system has reached some conclusion about the source of the observed disturbances, then this information and related information (e.g., explanation) is contextually relevant. Rather than force the operator to assemble step by step the relevant views and viewports, the system functioning as a cognitive tool would provide immediate access to a coordinated set of views and data needed to inspect, evaluate and follow up the intelligent system's assessment.

The proliferation of windows can be taken as a symptom of a design failure in the organization of different types of information according to different task contexts. Figure 5-6 shows a case where the operator must compare the control actions taken by the IS (based on a model of the MP) to the current state of the process and to the current goal (setpoint). When system parameters are out of tolerance, for example, the IS changes other parameters to maintain the goal. However, to track system behavior and distinguish normal process tuning from more severe faults, the operator may need to open three window to find and integrate the necessary data.

5.3 Message Lists and Timeline Displays

As figure 1-3 illustrates, one aspect of human-intelligent system cooperation is the set of mechanisms that help the human track the intelligent system's assessment, recommendations or actions with regard to the monitored process. The primary representational window used for this purpose in many of the NASA cases, as well as in other similar applications (Gradient project; Hollnagel, 1990), is a message list. Figure 5-7 shows one example abstracted from the case study. (Note that the following examples are abstractions from the observed cases, and do not use the actual text used in any particular system.)

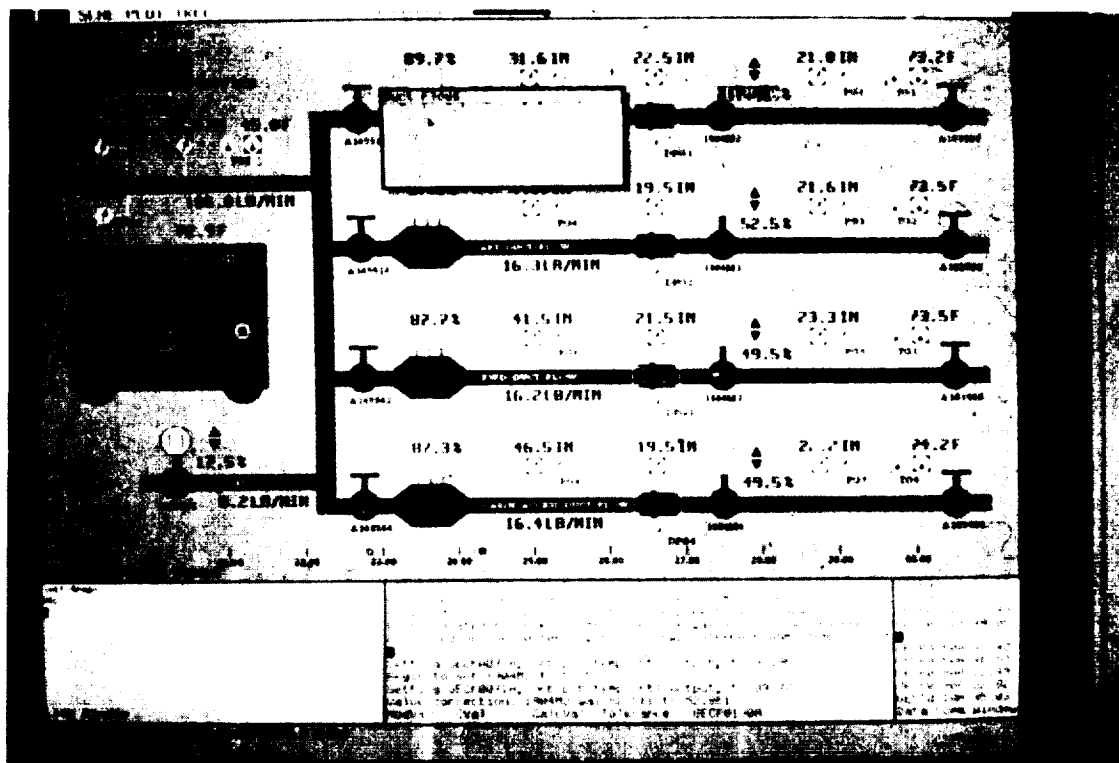


ORIGINAL PAGE IS
OF POOR QUALITY

The window in the bottom right of this workspace is a "scratchpad" window, and functions like the "clipboard" on the Apple Macintosh. High-level information from other windows can be copied into this space and thus presented in parallel with any other window.

Figure 5-5. Scratchpad Window the Functions Like Clipboard on Apple Macintosh®

® Macintosh is a registered trademark of Apple Computer, Inc.

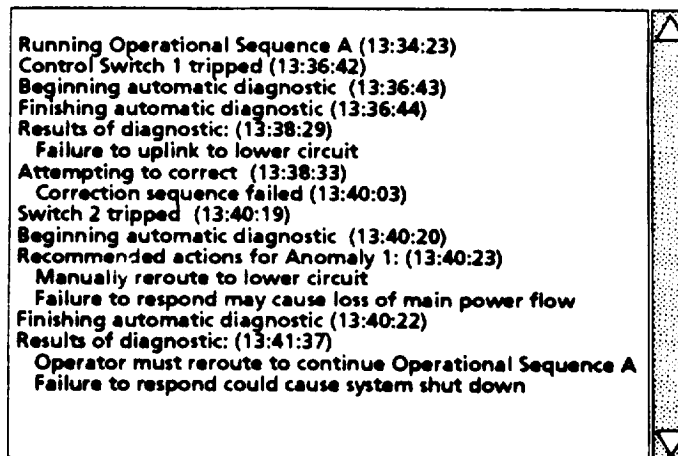


This example illustrates how overlaying the default window can obscure information. Clicking on a component opens a new window which is placed on top of the component and the digital value indicating flow rate.

This example also illustrates how a proliferation of windows indicates a design failure. When system parameters are out of tolerance, for example, the IS changes other parameters to maintain the goal. However, to track system behavior and distinguish normal process tuning from more severe faults, the operator may need to open three separate windows to find and integrate the necessary data.

Figure 5-6. Example Illustrating (1) How Information can be Obscured when a Window Overlays the Default View and (2) How Proliferation of Windows Indicates Design Failure

Full Message List window with Times



© 1991 Woods, Potter, Johannesen, Holloway

Figure 5-7. Example of a Message List Window with Timestamps, Messages and Format
Abstracted from Case Study

Message lists are windows that list events in a textual, alphanumeric string format. Typically each entry is one or two lines long. A timestamp is usually associated with each entry, but we observed cases where no timestamp was used. Message lists occurred in several different forms in the case study and included information about the monitored process (e.g., alarms) and automatic actions, as well as the intelligent system's assessment of the monitored process (see figure 1-4). While message lists were relied on heavily in the current versions of systems, most of the projects recognized weaknesses in this type of display and planned to develop improvements in future work.

5.3.1 Weaknesses of Message Lists

The exclusive use of textual encoding of system/intelligent system status means that the language of these messages is crucial to operator comprehension. It is important that it not be cryptic or redundant. Messages that are products of program execution traces may be particularly susceptible to this malaise. In addition, we saw several cases where the messages tended to "run together" making it hard to segment the list into individual messages.

Message lists usually operate on the principal that the last message is the most recent one. But there are no quickly apprehensible cues to indicate when the last message occurred -- messages that differ considerably in time of occurrence all look the same. The observer must focus his or her attention, read the timestamp and compare it to current time. The critical landmark of now, the current time, frequently is not included in the message list. "Time-less" message lists exacerbate the difficulty.

A message list organizes events chronologically. However, because each entry is a line of text, it is a "packed" representation that obscures the temporal distances between events. One cannot see at a glance whether events occurred contiguously or farther apart. Again it is only available through a deliberative cognitive process based on reading and comparing timestamps. Compare the two hypothetical message lists illustrated in panels A and B of figure 5-8. The same events occur in the same order in the two panels. What is different? Panels C and D of this figure immediately reveal the pattern of temporal relationships hidden in the packed list representation.

Because each entry is a line or lines of text, when a series of events occurs, the list of messages can exceed the space available in the viewport. This forces the operator to read and scroll through several screens of messages to see the inter-related messages and to "piece together" a global view of what is happening with the process, the pattern of automatic actions and the intelligent system's assessment or recommended actions (although the survey revealed cases where there was no scrolling capability, at least at that stage of development of the prototype system). Timestamps do not help the operator quickly apprehend this information; each message and timestamp must be read and compared to other messages. Furthermore, as one scrolls around in a long message list it is easy to get lost as the packed textual field contains no readily apparent temporal landmarks (e.g., current time was rarely indicated on the message lists surveyed).

Message List vs. Timeline Information Display for Two Sample Cases

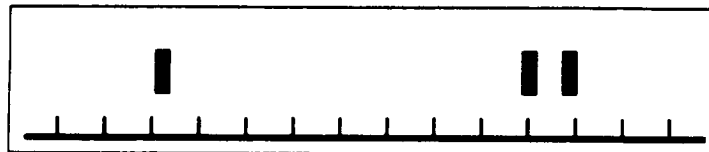
A.

10:23:43 Switch 3 tripped
10:31:07 Switch 4 tripped
10:32:23 Load Shed

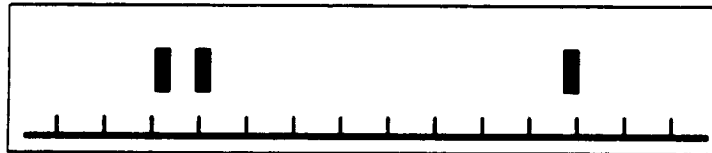
B.

10:23:43 Switch 3 tripped
10:24:27 Switch 4 tripped
10:32:23 Load Shed

C.



D.



© 1991 Woods, Potter, Johannesen, Holloway

The same events occur in the same order in panels A and B, but the timestamps are not enough to make the pattern of temporal relationships stand out. By indicating events against an analog timeline, panels C and D immediately reveal the temporal pattern.

Figure 5-8. Example of Sequence of Events That Points Out How Message Lists Obscure Information About Temporal Distance Between Events

Figure 5-9 provides an example (abstracted from the case study) of how relationships between messages can be very difficult to extract. The right panel shows the full message list. The left panel contains a smaller viewport with a portion of the list out of view. Messages concerning Switch 1 ("Switch 1 tripped" is referred to as "Event 1") have scrolled out of view in the left panel. The message that Switch 2 has tripped is displayed followed by the diagnostic message, "Recommended actions for Event 1: manually re-route to lower circuit". Because a portion of the list is hidden and because of the poor wording of the messages, this and the other diagnostic messages seem to be referring to the Switch 2 event when they are actually referring to the Switch 1 event. The message list makes it difficult to establish the context for each message 'at a glance'.

The above examples start to reveal how temporal patterns in event sequences are not readily apparent from a message list. The top panel of figure 5-10 shows a sequence of events against an analog timeline. The bottom panel shows some of the types of temporal patterns that the operator should be able to see. First, note that some of the events are unrelated to the later sequence -- independent subsets of events need to be discriminated. The sequence illustrates three types of temporal relationships important to diagnostic reasoning. An earlier event might be a premonitory sign of later major trouble. A single underlying fault may trigger several contiguous effects or have multiple contiguous manifestations (the event set labeled major event). The underlying fault will produce a cascade of disturbances and their associated manifestations which will be seen as a temporally evolving series of events that follow the original burst of activity (cf., figure B-1 and the related discussion in appendix B). In this case the diagnostician needs to see the cascade, distinguishing between manifestations that indicate the source of the trouble (the subset of events labeled major event) and those that result from disturbance propagation. In addition, the intelligent system may produce commentary about the events in the monitored process that are linked to different subsets of the event sequence.

One should note from the above that a message list is made up of different kinds of messages. The message lists surveyed generally did not distinguish between message categories. Figure 5-11 shows the uncategorized message list from the example mentioned earlier (figure 5-9). Figures 5-12 and 5-13 show two different ways to organize the same sequence of events. Figure 5-12 spatially segregates messages about two different anomalies. Within this categorization, the sequence of events should distinguish the anomaly, intelligent system (IS) diagnostic assessments, and automatic and recommended actions. The analog timeline and categorization of messages greatly enhances an observer's ability to track monitored process behavior and intelligent system assessment/actions. Figure 5-13 illustrates a different top level categorization by spatially segregating messages into categories of anomalies/automatic actions, IS diagnostics, and recommended responses. Again, combining the analog timeline and categorization reveals the pattern of relationships in the sequence of events and fulfills the design goal illustrated in figure 1-4.

An extension of these ideas that supports message comparison is two side-by-side message displays, each containing one highlighted message. A horizontally-oriented timeline is above them, highlighting them amongst the other events and giving an analog perspective on the temporal distance between them. Under both of them is (1) the duration between the message and the current time and (2) the duration between one message and the other.

Some of the systems surveyed did try to categorize messages by placing different types into different message list windows. However, the lack of indication of temporal interval and the packing of messages removes any anchor for comparison across windows (see figure 5-14). Furthermore, the workspace design provided the user with window placement flexibility creating the opportunity for nonalignment of the related windows. (Actually, in this case it may be even more misleading to align the windows since there is no necessary temporal ordering relation maintained across the windows.)

Partial Message List window

Switch 2 tripped
Beginning automatic diagnostic
Recommended actions for Anomaly 1:
 Manually reroute to lower circuit
 Failure to respond may cause loss of main
 power flow
Beginning automatic diagnostic
Finishing automatic diagnostic
Results of diagnostic:
 Operator must reroute to continue Operational
 Sequence A
 Failure to respond could cause system
 shut down

Complete Message List

Running Operational Sequence A
Control Switch 1 tripped
Beginning automatic diagnostic
Finishing automatic diagnostic
Results of diagnostic:
 Failure to uplink to lower circuit
Attempting to correct
 Correction sequence failed
Switch 2 tripped
Beginning automatic diagnostic
Recommended actions for Anomaly 1:
 Manually reroute to lower circuit
 Failure to respond may cause loss of main
 power flow
Finishing automatic diagnostic
Results of diagnostic:
 Operator must reroute to continue Operational
 Sequence A
 Failure to respond could cause system
 shut down

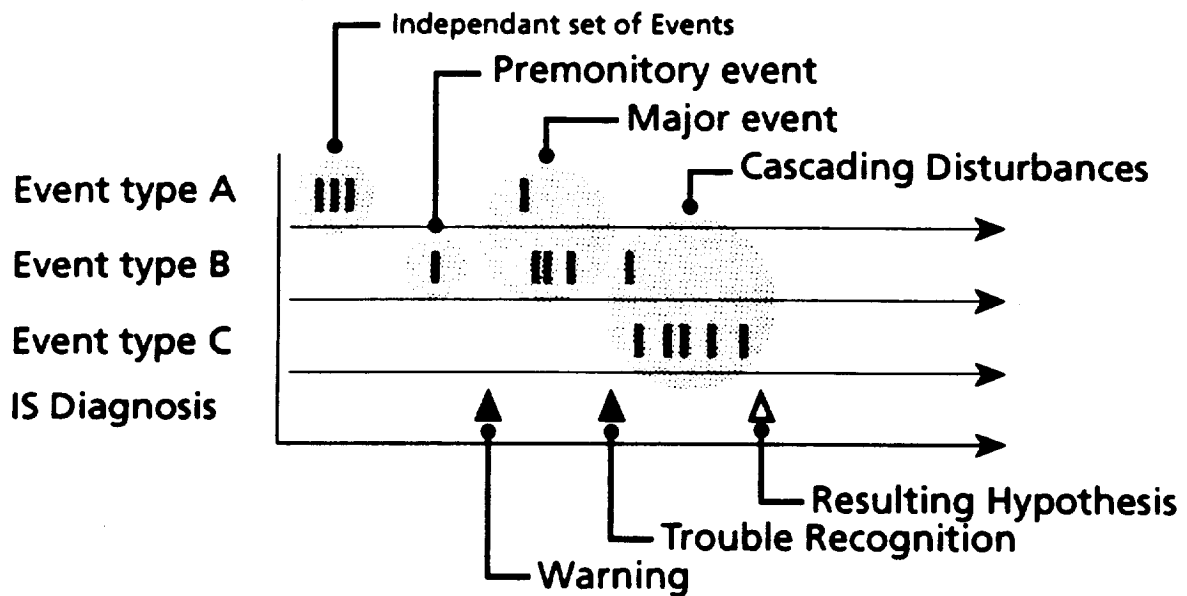
© 1991 Woods, Potter, Johannesen, Holloway

Figure 5-9. Complete Message List and Partial Message List Containing Same Messages, Some of Which Have "Scrolled Away"

Unorganized Sequence of Events in Time Line Format



Patterns of Events



© 1991 Woods, Potter, Johannesen, Holloway

Above: Sequence of events spatially displayed on a timeline conveys temporal relationships.

Below: In addition to spatio-temporal information, event-type categorization information allows one to see the pattern of events.

Figure 5-10. Relationships Revealed by Displaying Events on a Timeline

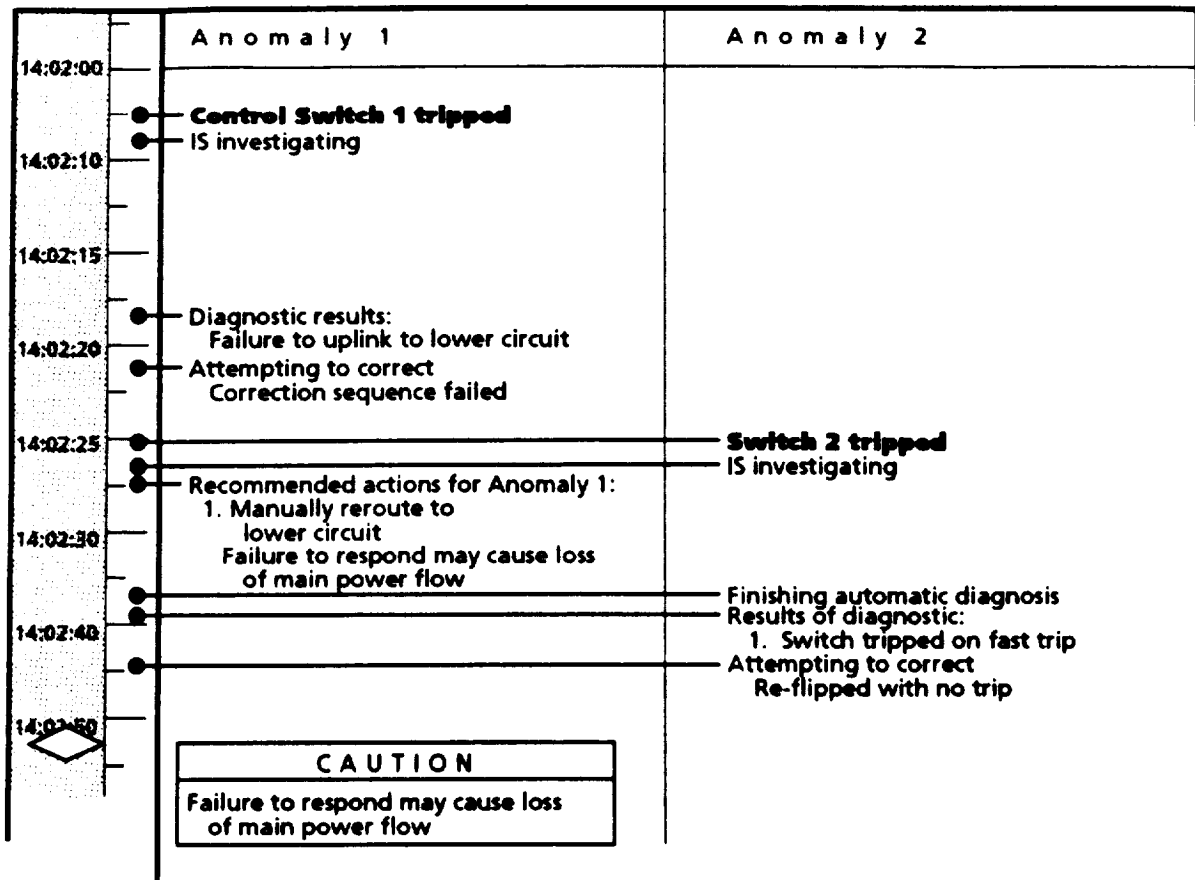
Panel A

<p>Control Switch 1 tripped Beginning automatic diagnostic Finishing automatic diagnostic Results of diagnostic: Failure to uplink to lower circuit Attempting to correct Correction sequence failed Switch 2 tripped Beginning automatic diagnostic Recommended actions for Anomaly 1: Manually reroute to lower circuit Failure to respond may cause loss of main power flow Finishing automatic diagnostic Results of diagnostic: Switch tripped on fast trip Attempting to correct Reflipped with no trip Failure to respond to Anomaly 1 could cause system shut down</p>	<p>△</p> <p>▽</p>
--	-------------------

© 1991 Woods, Potter, Johannesen, Holloway

Figure 5-11. Typical Message List

Panel B



© 1991 Woods, Potter, Johannesen, Holloway

Figure 5-12. Messages Shown Categorized by Anomaly Type

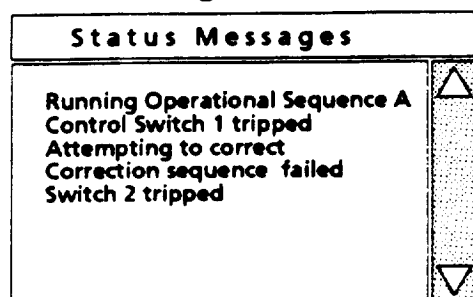
Panel C

	Automatic/Anomaly Actions	Diagnostic Messages	Recommended Responses
9:57	Beginning boot-up		
9:58	Beginning automatic diagnostic		
9:59			
10:00	System review complete		
10:01	Main program logged in		
10:02	Boot-up complete		
10:03	Operational sequence A selected		
10:04	Operational sequence A booting		
10:05	Running Operational sequence A		
10:06			
10:07	Control Switch 1 tripped	Failed to up link to lower circuit	<div>Manually reroute to lower circuit</div> <div>Caution</div> <div>Failure to respond to Control Switch 1 may cause system shut down</div>
10:08	Switch 2 tripped	Switch 2 tripped on fast trip Re-flipped with no trip	
10:09			
10:10			
10:11			
10:12			
10:13			
10:14			

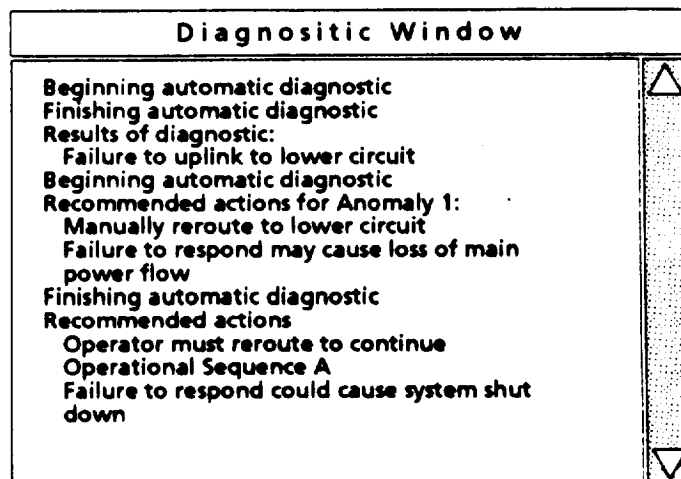
© 1991 Woods, Potter, Johannesen, Holloway

Figure 5-13. Messages Shown Categorized by Type of Message: Automatic Action, Diagnostic Message, or Recommended Action

Status Message List window



Diagnostic Message List window



© 1991 Woods, Potter, Johannesen, Holloway

Note that there is no temporal scale relating the two windows.

Figure 5-14. Messages Categorized by Type into Two Separate Windows: Status and Diagnostic

Figure 5-15 depicts another attempt to categorize messages into different message list windows, in this case, into two windows one with messages in an abbreviated textual format and the other as an "explanation" of the events coded in the first list. If the operator is confused by the abbreviated format he or she must switch back and forth between the two windows. The operator must figure out which explanation goes with which message. The timestamps provide the only means of calibration between the two message lists. Note that not all messages in the status window have an explanation. Relative position is a misleading cue since there are no time markers between the lists; for example, the last explanation does not necessarily correspond to the last (most recent) message in the status list.

The above two examples illustrate a theme referred to previously as the dissociation of related data. The separation of related messages imposes extra cognitive tasks on the operator to find, collect and integrate the information on his own.

Overall, the problem with message lists is that it imposes cognitively effortful deliberative process of finding, collecting and integrating individual messages in order to construct the meaningful relationships and patterns between events.

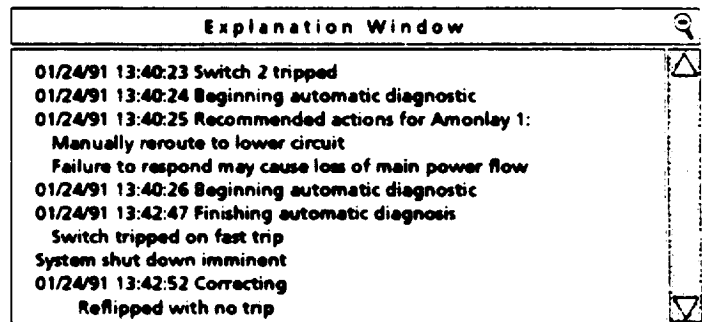
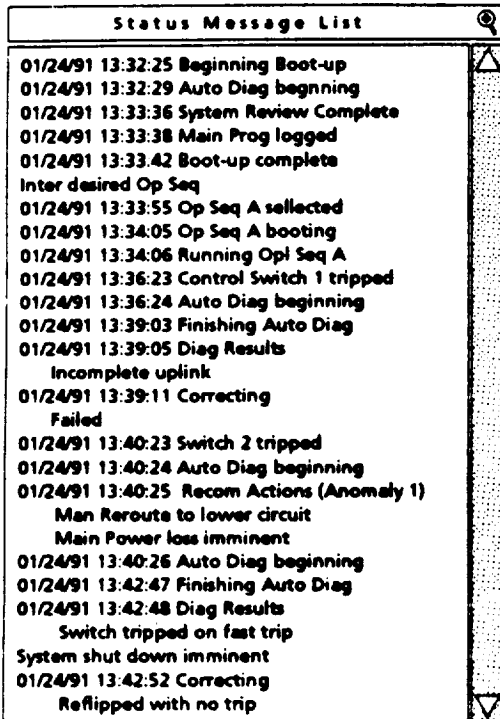
The question that remains is: what alternative representations can be crafted to overcome the deficiencies of message lists? The first step is to recognize that message lists are just one instance of temporally organized displays of information. Another type of temporally organized display is a timeline where an analog representation of time is used as the organizing anchor to represent sequences of events. In fact, some of the cases did reveal timeline representations, most notably in the Intelligent Launch Decision Support System. What is interesting is that in these cases timeline representations were developed based on the existence of a planned sequence of events -- what could be called Plan Based Timelines (see figure 5-16). Message lists can be replaced by what could be called -- Event Driven Timelines (see figure 5-16; figures 5-11, 5-12 and 5-13 are also examples).

5.4 Physical Topology Schematics

One aspect of human interaction with an intelligent system in dynamic fault management situations is the representation of the monitored process (refer to figure 1-2). One of the kinds of process views that occurred frequently in the case study was a schematic diagram of the monitored process. This type of display can be called a physical topology schematic because the organization of the graphic is based on the physical topography of the process -- the subsystems, components and their physical interconnections. Active data about the state of the monitored process -- parameter values or component states -- are annotated to the schematic.

Physical topology schematic diagrams or simply 'physical schematics' are relied upon by many systems to display an overview of the monitored process. These physical schematics tend to be reproductions of paper-based schematic diagrams, with certain modifications. It is important to be aware of the implications that the following modifications have on operator information extraction.

One important modification is that the VDU-based physical schematic incorporates dynamic data; parameter values are placed physically next to, or on top of, the representations of the components or sensors, etc, associated with them. The schematic is effectively given a dual task: rather than simply provide a view of the physical ins and outs of the system, it is intended also to provide a living picture of the state of the process.



© 1991 Woods, Potter, Johannesen, Holloway

Seeing what status message relates with what explanation is not easy in this example.

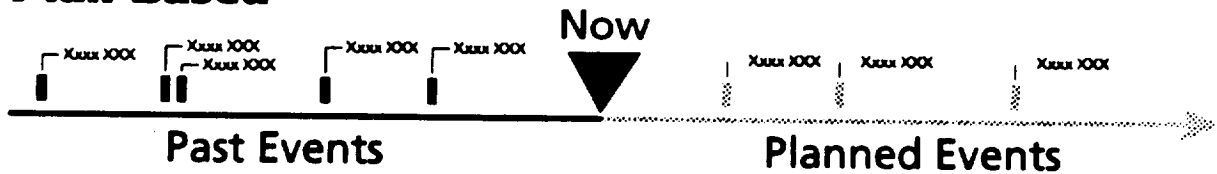
Figure 5-15. Messages in Status Message List "Explained" by Messages in Explanation Window

Timeline Formats

Event Driven



Plan Based



© 1991 Woods, Potter, Johannesen, Holloway

Above: Event-Driven format to show past events spatio-temporally organized. Below: Plan-Based format to show both past and planned events in a spatio-temporal organization.

Figure 5-16. Comparing Event-Driven Format to a Plan-Based Format

The other important modification is imposed by the nature of the host medium; real estate on a VDU is precious and so the schematic may undergo a simplification of detail. Furthermore, the VDU schematic must be viewed in a different way from the way its paper counterpart can be viewed. For example, techniques such as zoom and pan can be used to move a viewport over a large schematic.

In order to design an effective view of the monitored process it is imperative to define the information that operator needs in order to perform his tasks, i.e., what are the information extraction goals for this display? A physical schematic of the monitored process is useful only if the operator needs to know about the different physical parts of the system and their interconnections. For example, knowing where a sensor is located relative to other sensors/components may be important in extracting the significance of its current reading or recent behavior. But physical schematics are often used when the information transfer goal is to show the state of the monitored process -- is it working correctly? is it achieving its goal? State assessment cognitive tasks may be much better supported by functional or model based displays (Mitchell and Saisi, 1987).

Some of the problems observed in the physical schematics surveyed include: (1) poor perceptual segmentation, (2) an overreliance of digital representations to indicate state information, (3) the lack of a functional model to organize and integrate data, and (4) the lack of any integration of measured data on the state of the process with the intelligent system's assessment.

It seems a truism to say that what is important should stand out; the operator should not have to expend effort in finding out needed information. Change can be generally said to be important and should be conspicuous. Yet we have observed schematics and other display forms where the most perceptually salient feature was a static, even unnecessary, element, such as the name of the system or of the organization. In general, the physical topology schematics failed to help the operator see the patterns of events and quickly size up the dynamical state of the monitored process. When coupled with the problems produced by the proliferation of windows and message lists as the primary form of output from the intelligent system, the result is inscrutable systems -- the wall illustrated in figure 1-2.

The over-reliance on digital/linguistic forms of representation resulted in the dissociation of related data, failure to highlight anomalies, and failure to highlight state changes and dynamic behaviors in the monitored process. In one system a physical schematic served as the main window on the screen (figure 5-17). The schematic shows digital parameter values which are dynamically updated. Because operators had trouble seeing changes in the monitored process when inspecting the schematic, another smaller, scrollable window was created in the corner of the screen to show the most recent parameter value or component state changes as a message list.

The second window attempts to deal with the problem of calling the operator's attention to changes in values that occur, but which are not readily perceived in the schematic. While it is easier to detect new values in the smaller window there are several reasons this is a poor strategy for presenting dynamic information. First of all, data is being repeated. It should only appear once, effectively displayed. Also, it takes up screen real estate. More importantly, though, having the data in two windows forces the operator to switch back and forth among views in order to integrate them. This "dissociation of data" is inefficient and effortful. Also, note that rate of change information for the component values is not available even in the small window. Finally, we can point out that the small window is problematic in and of itself, being an instance of a message list format (the problems with this format are discussed in another section).

ORIGINAL PAGE IS
OF POOR QUALITY

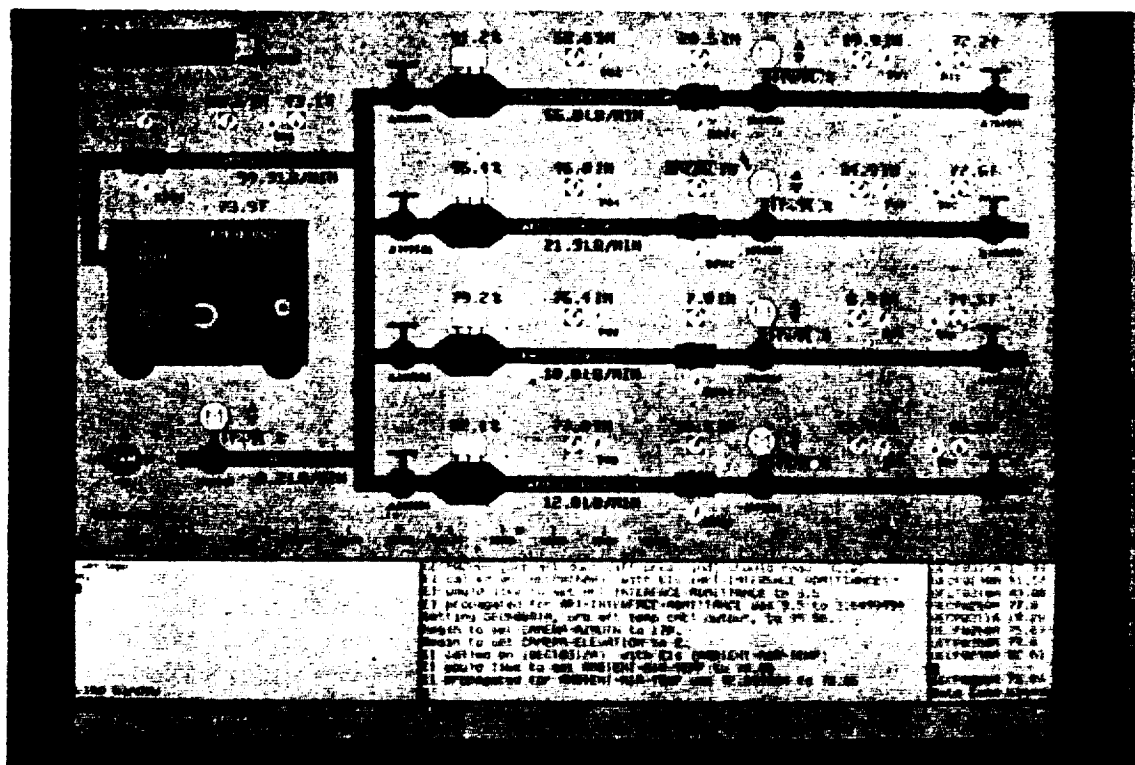


Figure 5-17. Parameter Value Updates in Schematic Must be Repeated in Messages Window at Right Because They are not Easily Perceived in Schematic

In an attempt to present a "clean" schematic display some detail may be sacrificed, to the detriment of the operator's information needs. In some cases the detail is shifted to another window, but the movement among displays may be distracting and effortful, particularly when it is not natural or obvious where to find information. In some cases the needed detail may be left out of the workspace altogether, i.e., the schematic may contain inaccuracies (unfortunate, and perhaps rash sacrifices to the real estate god, as it were). For example, in one system surveyed, the intelligent system performed certain controlling functions, but these were not displayed to the operator; only a simplified, incomplete schematic version of the process was displayed. The information that was missing was necessary for forming an accurate picture of the process. It is crucial to make the intelligent system's actions on the process visible in order to give the operator the most accurate model of the process and to allow the operator to detect any problems or limits of the intelligent system.

An example of a display that attempts to show an overview of the state of the monitored process but fails to do so because it does not represent the process functionally is shown in figure 5-18 ("Status at a Glance"). This display provides dynamically-updated, raw sensor values in a digital format annotated to a abstracted representation of the physical topology of the system. Below it in the figure is shown the paper-based physical topology schematic from which the display was derived.

Note that the values of the different components are not set within the context of expected values or limiting values. This means means that the operator must remember this information in order to interpret the significance of the displayed parameter values. Also, the data displayed is not integrated in an informative way. For example, the relationship between temperature and pressure is not made explicit; this relationship is needed to monitor subcooling margins, for example. The display forces the operator to search through the data and makes him perform the extra task of information integration. Information extraction in order to evaluate the process state is made effortful and inefficient because relationships, and patterns of data are not available. A study was recently done that indicated the ineffectiveness of this particular non-model based display (see Czerwinski, 1991.) A model-based view of the monitored process given by a functional schematic would allow the operator to see the functional relationships in the process and thus extract the critical data effectively.

5.5 Wading through the Interface Layer

5.5.1 Dissociation of Related Data

The following example serves to illustrate the extra workload that can be imposed on the operator when one piece of information is parceled out throughout the interface layer.

In one system we observed, parameter values are displayed digitally on the screen and color coding is used to indicate the state of these parameters: white means normal, red means the component is being tested, and purple means a diagnosis has been performed and the component is in some sort of abnormal condition. However, the precise meaning of the abnormality is available on a different display. Figures 5-19 and 5-20 illustrate the stages involved in becoming informed about an anomaly in a generalized version of this system.

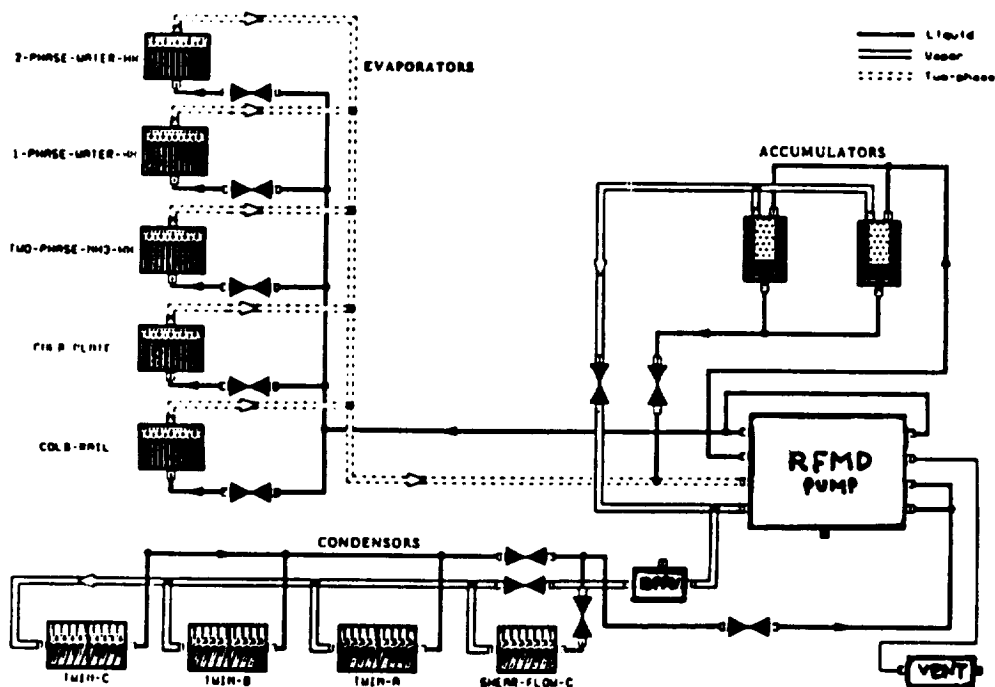
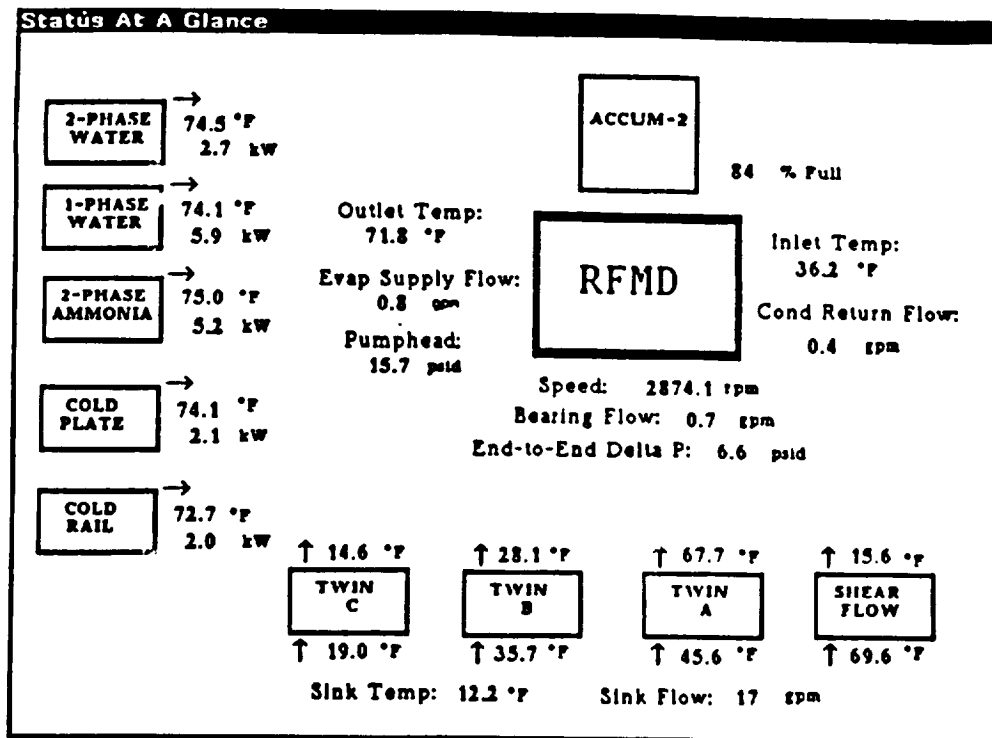


Figure 5-18. "Status at a Glance" Display and Paper-Based Physical Topology Schematic from Which It was Derived

POSSIBLE SEQUENCE IN BECOMING INFORMED ABOUT AN ANOMALY
(An Example)

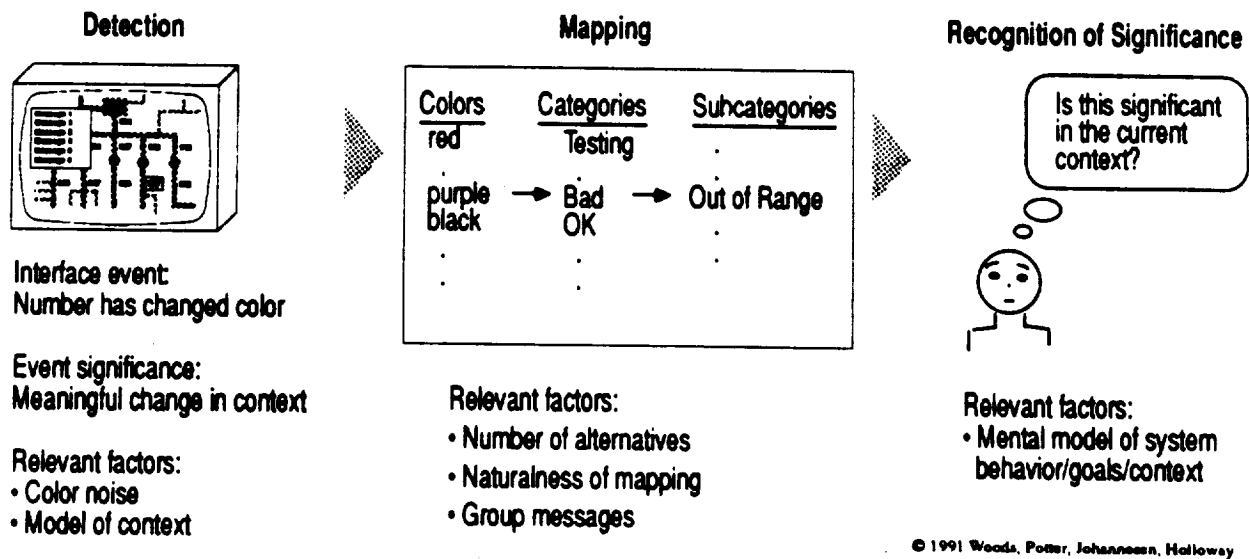
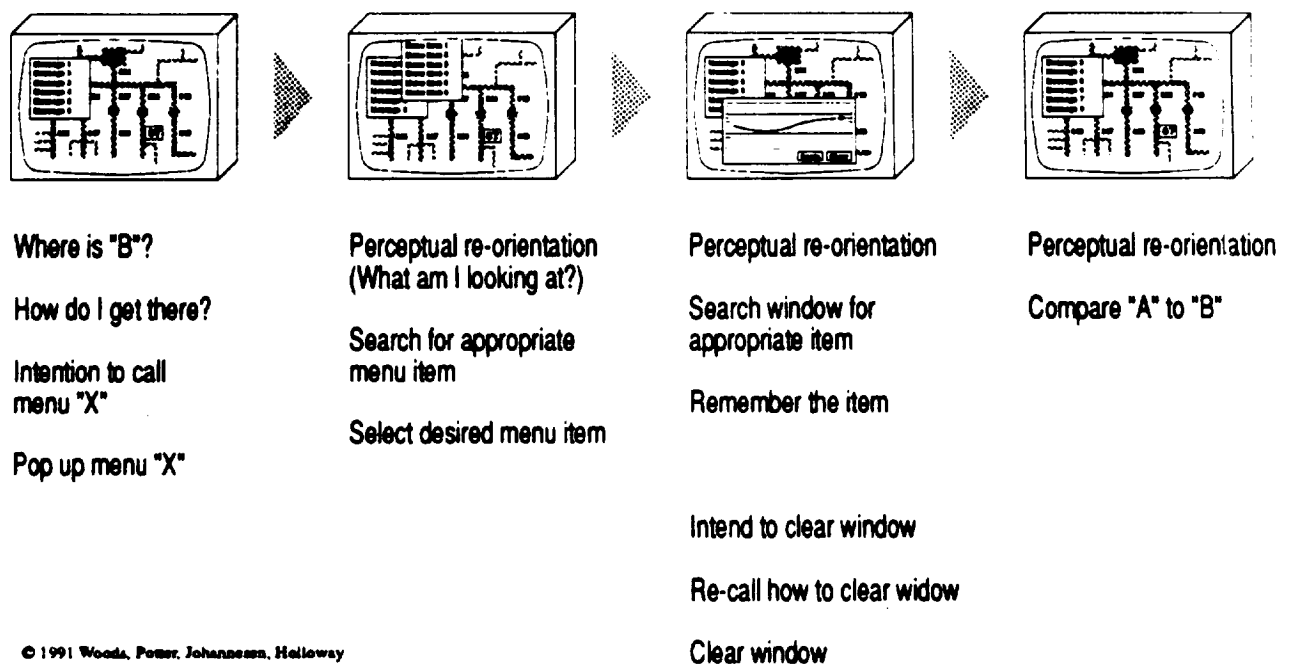


Figure 5-19. Stages 1-3 in Becoming Informed about an Anomaly

POSSIBLE SEQUENCE IN BECOMING INFORMED ABOUT AN ANOMALY
(An Example)



© 1991 Woods, Peter, Johannessen, Halloway

Figure 5-20. Stages 4-7 in Becoming Informed about an Anomaly

When the system diagnoses an anomaly, a number on the screen will change to purple. The operator's ability to detect the color change depends on the backdrop that is present. Questions relevant to the change's detectability are: how much color noise is there? how many other items on the screen are of the same color? how much data is present? Also, if the operator was not looking at the screen at the moment the change occurred, will he be able to tell a change has occurred? Because no time information on the changes is immediately available on the screen, it is possible for the operator to slip and think he knows about recent changes when he does not.

Assuming that the operator detects the change to purple of some parameter value, the operator would then mentally map the color to its generalized meaning, i.e., that purple means an abnormality. (If there were several alternative color changes possible, each with a generalized meaning, this would affect recall of the meanings of colors.) In this particular case, however, there are only three.

Then the operator must make a decision about whether this change is important in the particular situation, i.e., whether it is worth retrieving the specific information on the kind of abnormal condition being referred to by the color change. At this stage the operator's mental model of the monitored process and intelligent system comes in to play. Did some action prior to the color change allow him to expect that the component would be faulty, so that he wouldn't think it necessary to seek out the actual diagnosis?

If the operator decides he needs to find out more about what the specific abnormality is, he then needs to try to get to the appropriate display. The relevant issues here are whether 'getting to the appropriate display' is visible on the interface, or, if it is not immediately visible, whether the sequence of events is natural and memorable. If it is complicated, perhaps involving a couple of menus with vague menu items, a memory load will be imposed. If the operator cannot remember the sequence, a workload burden will be imposed in trying to find out the sequence.

Once the operator executes the commands to call up the appropriate display or window, the operator must "orient" to it, i.e., he must search for the relevant data. The workload associated with this task depends on the other data in this view and the organization of the display. When the operator has retrieved the target data, he may need to clear the viewport which he had opened or de-clutter the workspace in order to return to the previous, 'overall status' display. Again, the naturalness and the sequence of commands required to do this affect the workload. Once the operator returns to the previous display he must reorient to it. For example, he must compare his mental model of the current events in light of the diagnosis with what is on the display and to establish whether any other changes have occurred.

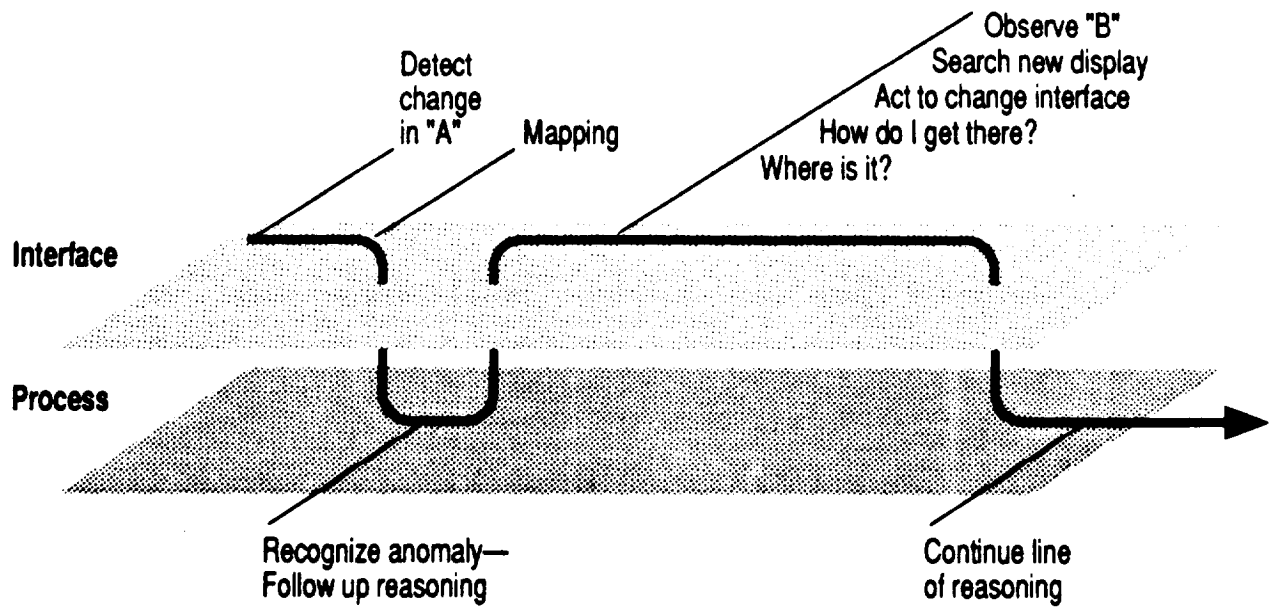
This parceling out of the diagnostic information is inefficient; it burdens the operator with extra tasks in order to acquire related data. Figure 5-21 illustrates the points concerning the potential lack of transparency of the interface layer and the effortful task it can be to travel through it.

5.5.2 Hidden Functionality

Some of the systems had hidden interface functionality i.e., the system may have powerful capabilities but what these are they are how to use them is not apparent from the screen.

In most of the schematic interfaces we observed, it was not obvious what the available choices were at any point. This lack of visibility concerning what to do to get what you want is what Norman terms the 'gulf of execution.' The functionality of the interface (i.e., what it is possible for a user to do) should be natural and should not impose a memory requirement.

POSSIBLE SEQUENCE IN BECOMING INFORMED ABOUT AN ANOMALY
(An Example)



© 1991 Woods, Potter, Johannesen, Holloway

Figure 5-21. Wading through the Interface

Many interfaces have mouse sensitive icons that when clicked provide more information on the item. For example, in one system, the sensors and switches are mousable. The problem here is that it may not be clear what is selectable and what is not. Also, it can be confusing, if the same functionality (e.g., mouse-click) can produce an operation on the user-interface in one mode of operation, say, and an action on the monitored process in some other mode.

5.5.3 The Problem of Fleeting Data

A classic example of the problematic effects of fleeting data is illustrated by what occurred at the Electrical, Environmental, Consumables, and Mechanical Systems (EECOM) console during the Apollo 13 mission. Just a few seconds before the oxygen tank ruptured, the oxygen pressure value on the screen increased very quickly until it soared to 1008 p.s.i. Three seconds later it plummeted to 19 p.s.i. The EECOM's attention, however, was directed towards the Hydrogen tank pressure, the measurement of which had been giving them problems since the beginning of the mission (Cox and Murray, 1989). Consequently the EECOM did not notice the rising and then falling oxygen tank pressure. If he had seen the behavior of the values, he would have been able to deduce the tank had ruptured. However, this information was lost as soon as the values changed. A timetail plot of this data would have captured the behavior of the oxygen tank pressure and would have been very useful in this case.

This problem of fleeting data is not limited to parameters changing on a display. In one of the cases observed, the scheduling component of the system scheduling system generates a new schedule of activities if, for example, a power source is lost. However, the user is not informed of how the updated schedule differs from the previous schedule. Too much reliance is placed on memory in being able to detect changes.

In one system we observed, icons are used to represent switches. When the switches are being tested by the expert system, the icons flash red and then either turn black, if the switch is faulty, or remain uncolored if it is normal. The problem is that the information communicated by the flashing icon is lost if the operator does not happen to see it as it occurs. When information is fleeting as in this case there is potential that the operator will not know if or when it occurs. This is another factor that contributes to an inscrutable system, reinforcing rather than breaking down the barriers shown in figure 1-2.

5.5.4 Overuse of Digital Representations and Underuse of Analog Forms

Analog representations can be used to provide the contextual data that gives significance to a given datum (cf., Woods and Elias, 1988). Parameter values often need to be presented within the context of their limiting values and of their expected values. Even digital representations can be enhanced to show contextual data to some degree. In one system surveyed, a display consisted of a table of telemetry channel values in digital form. Alongside each current value of the channel was the expected value and the range of possible values.

Section 6

The Development Process

The development process was initially investigated to identify how to effectively deliver the design guidance discussed in sections 4 and 5. Case study observations of the development process, however, identified a more fundamental design problem than delivery of guidance. It was observed during the case study that perceived user interface design problems are often actually intelligent system design problems (e.g., unavailable information, mis-representation of information). These problems result from failing to identify the information exchanged between human and computer when performing domain tasks (i.e., information requirements). These information requirements are defined as a part of HCI analysis and design.

Although HCI design can affect the entire support system, including conventional software systems, intelligent systems, and the user interfaces to both, it is usually not considered during system design. At best, the user interface design is influenced by HCI considerations. The user interface primarily addresses presentation *media* (i.e., presentation of information), however, and is not concerned with the information *message* (i.e., information content) that is also a part of HCI design. Since user interface design is frequently done after system design, the potential for HCI design to influence system design does not exist. It is necessary to modify the design process to include definition of the information requirements for both the support system (both conventional and intelligent software agents) and the user interface.

This process view of system design is often absent during intelligent system design. Designers tend to focus on specific technology aspects of the design (e.g., what software tools will be used) and fail to consider the overall system view of the design (e.g., how will the intelligent system be integrated into the existing support environment). Most types of design guidance address technology issues instead of process issues (Shafto and Remington, 1990). A development methodology is needed that addresses design process issues.

Development methodology seems to be the most effective means of integrating HCI design into system design. The methodology should support mapping from a specification of the task into information requirements for both the intelligent system and the user interface. This task description should include both domain tasks and tasks required to coordinate human and computer agents. The methodology should incorporate both information-level and display-level design guidance as an integral part of design. It should provide mechanisms for communication and coordination between members of a multi-disciplinary design team. It should support development of the intelligent system as an integral part of the overall support system, which may include both intelligent and conventional software. Each of these topics are discussed in greater detail in the subsequent sections.

6.1 Development Methodology

Many of our insights into the steps for developing information requirements result from a new perspective of the user -- the user as another type of agent in a heterogeneous, cooperating, distributed system. System design then becomes the design of an architecture for accomplishing domain tasks with the available human (i.e., users) and computer (i.e., applications) agents. HCI considerations are an important part of such system design, even before user interface design is addressed.

A significant result of this project is the identification of how HCI design and analysis affects design of both the intelligent system and its user interface. To understand these effects, one must understand how information requirements are derived from a specification of the task. These information requirements impact all elements of the system (i.e., application, user

interface, user). A design methodology for developing information requirements should include the following steps:

- **Description of domain tasks**

The fault management task is described in terms of goals and the actions required to achieve these goals. This task description should include actions required to coordinate agents during joint tasking.

- **Identification of resources provided to perform tasks and the constraints that affect task performance**

Resources include the capabilities of the operational environment and of the agents, and the availability of information. Constraints result from complexity, dynamics, and deficiencies in these resources.

- **Specification of agent activities and valid agent behaviors in an architecture for multi-tasking and dynamic task allocation**

Actions defined in the task description are assigned as specific agent activities consistent with the available resources and the inherent constraints of the fault management team and the operational environment. These agent activities support both fault management of the monitored process and agent coordination.

- **Evaluation of the activity specification using complex activity sequences and modes of agent interaction**

The specification of agent activities is evaluated using operational scenarios to derive the information and functionality requirements. These scenarios should include both nominal and anomalous activity sequences and modes of agent interaction.

- **Analysis of requirements for system information**

Requirements for information are explicitly identified for both the applications and the user interface.¹

- **Design application and user interface using information requirements**

Information requirements are applied in the design of the intelligent system, its user interface, and any conventional software used in the support system.

Figure 6-1 illustrates these steps in the development process that map from a task description into information requirements for both the intelligent system and the user interface

¹ - Functional requirements for the intelligent system may also be identified by these analysis tasks.

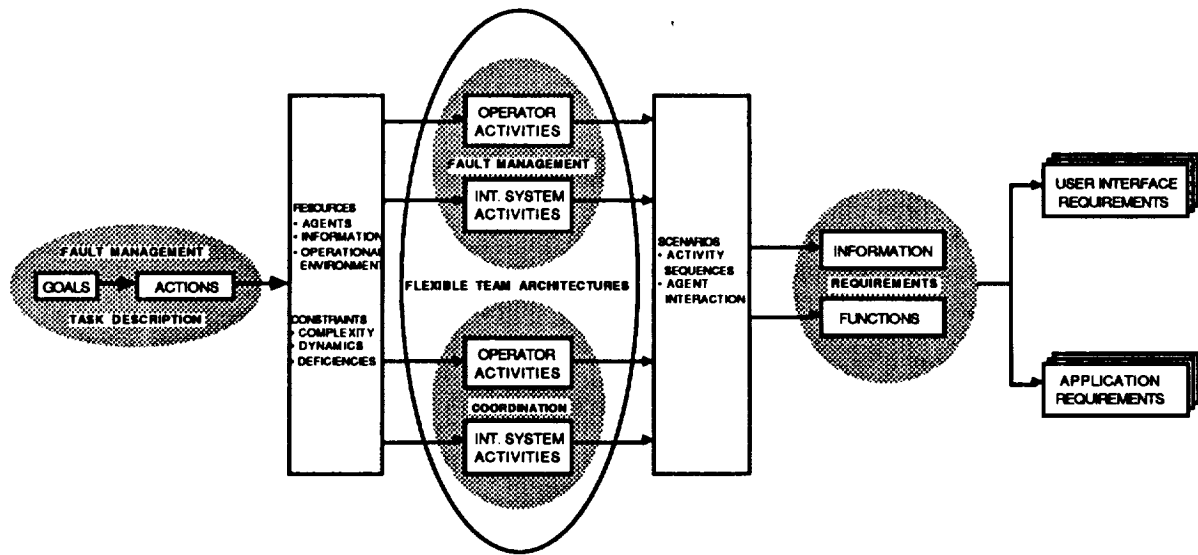


Figure 6-1. Mapping from Task Description to Information Requirements

This characterization of the design process outlines the steps of a development methodology. The methods and software tools that could assist designers in performing these steps have not been fully defined. Additional investigation is required to identify appropriate methods and tools and to integrate these techniques into a development methodology.

Issue: Methods and tools are needed to assist in mapping from the task description to information requirements. These techniques should be integrated into a system development methodology consistent with the development steps that have been defined in this section.

These steps define *what* needs to be done to develop information requirements. The designer must also understand *when* these steps are executed. Most applications in the case study were developed using iterative prototyping. The complete design methodology will consist of these steps integrated into an iterative development life cycle. In such a methodology, development would generally proceed from the task description (at the left of figure 6-1) to the information requirements (at the right of figure 6-1). It should be possible, however, to go back to previous steps as additional information is obtained (e.g., continuing knowledge acquisition may update and refine original task description).

In the remainder of this discussion of development methodology, the steps outlined above are elaborated and recommendations and issues associated with these steps are provided.

6.1.1 Task Description

Task description is defined as a specification of the task in terms of the goals and the actions required to achieve these goals. This task description differs from the traditional task analysis performed when designing the user interface. Evaluation of the development process has identified deficiencies in traditional task analysis for systems that have complex agent interaction and multi-tasking.

For complex, multi-agent systems, a significant number of activities can be classified as *coordination* activities, which are activities independent of the domain task that are required for

the human and computer to work together effectively (see section 3.3.2). Coordination activities act upon the agents responsible for domain tasks, while activities derived from domain tasks act upon the process being managed. The activities and information required to coordinate multiple, interacting agents are not identified by task analysis techniques yet are critical to the success of the system in operation. A design approach that specifies architectures for cooperation between human and computer agents is a promising way to incorporate coordination activities into system design (see also the discussion of dynamic task assignment in section 4.1.1). Task analysis techniques for use early in design should be modified and simplified to facilitate identification of these coordination activities, and to exclude details of user interface design by focusing on information requirements.

Most task analysis techniques represent tasks as sequences of low-level activities (e.g., individual keystrokes). For systems with complex agent interaction and multiple, simultaneous tasks, such an approach quickly becomes unmanageable. It is necessary to represent tasks at a higher level of abstraction. The task description should specify task goals and procedures and identify the information required to perform tasks.

Modified techniques for task analysis should assist designers in using the task description throughout the software development process. Typically, knowledge acquisition for intelligent systems is performed iteratively. A means should be provided for updating and maintaining the original task description as additional knowledge about the task is uncovered (Jones, 1991). Methods for evaluating the accuracy of the task description should also be supported (Bloom, 1991).

Issue: Modified techniques for task analysis are needed that result in a task description representing task goals, procedures, and information flow. These techniques should be usable early in the development process and should facilitate identification of coordination activities as well as fault management/domain activities. Support for modifying and maintaining the task description should also be provided.

If such techniques for managing and modifying the task description are provided, it can serve as a common repository for task information useful to the entire design team. As such, it can assist in organizing design team activities and serve as a means of communication between design team members about the task (see the discussion of design team later in this section). It may even be possible to reuse portions of the task description for new support systems with similar agent activities or derivative monitored processes (Jones, 1991). See appendix D for an example of a partial task description of aerospace fault management that has potential for reuse.

The representation of goals and procedures is useful not only in describing the task for use in system design, but can also be used explicitly in the design. The goal structure can be reproduced on the user interface as a useful means of communicating with the operator about the task (Mitchell, 1991). The goals and procedures currently being pursued also indicate the intent of operator actions. They represent a context for the interpretation of information and can improve collaboration between agents about the task.

6.1.2 Techniques

In this section, two techniques for developing requirements are described. The first technique, *prototyping*, is almost universally applied in the development of intelligent systems. The second technique, *storyboards*, are less commonly used, but proved very useful when tested in the development of the PDRS Decision Support System (DESSY).

Prototyping

Most of the applications in the case study were developed as iterative prototypes. Often, the prototype represented the final requirements. When the delivery system will be developed by a separate organization, the prototype must be translated to requirements that are amenable to explicit verification. Demonstration of capability corresponding to specific requirements is typically the mechanism used by a development organization to verify a software product.

Issue: Prototyping is frequently used to define requirements for a deliverable system. In addition to defining system capability, requirements also represent a means for the developing organization to certify that a software product provides the specified capability and to prove that contractual commitments have been met. Techniques are needed to translate a prototype into verifiable requirements.

There is a difficulty in using an iterative design process within NASA. The waterfall life cycle has been baselined for major programs such as Space Shuttle and Space Station. The waterfall life cycle, however, is not well-suited to the iterative design of software. Investigation of the design methodology should include integration of iterative design into the waterfall life cycle.

The frequent design changes resulting from iterative prototyping can be difficult to implement if the prototype is not designed properly. A small design change may translate to a large implementation change. It is especially important to separate the user interface from the intelligent system, so that changes in one will not impact the other. Many tools that support iterative design do not enforce such a separation. For example, many of the prototypes developed in G2™ freely mixed display rules with intelligent system rules. The prototype toolkit User-Intelligent System Interface Toolkit (UISIT) supports an architecture with separation of the user interface and intelligent system. UISIT is being evaluated as part of the effort to define requirements for the design methodology.

Issue: Iterative prototyping involves frequent design changes. These changes can be in either the user interface or the intelligent system. Design tools are needed that enforce separation of the intelligent system and its user interface to minimize the impact of changes in one design on the other design.

Preliminary designs in the form of rapidly-developed prototypes can be evaluated to elicit additional information requirements. A common method of evaluating a prototype is testing using operational scenarios. Exercising the system under realistic conditions allows the designer to identify missing or misrepresented information and cumbersome interaction sequences. Evaluation of designs using operational scenarios also represents a way to incorporate HCI design early into the development process.

Problem: Testing the HCI Design

Recommendation: The HCI design can be evaluated by testing a user interface prototype using expected operational scenarios. This evaluation should occur early in the design process and should assess such issues as availability of information and ease of interaction.

™ G2 is a trademark of Gensym Corporation.

One goal of operational testing is to determine if system behavior meets acceptance criteria. For such testing to provide realistic results, the data used for these tests should closely resemble the data actually used for flight support.

Problem: Testing the Design under Realistic Conditions

Recommendation: The prototype should be tested using high fidelity, simulated data or real data. If the prototype represents an upgrade to existing technology, such testing is most effective when both the existing system and the new system are used side-by-side (i.e., the term *parallel operations* is used by the RTDS developers).

The RTDS project provides an example of testing using real data. Workstations in both the Mission Control Center and the RTDS development lab have connectivity to real-time data from the integrated simulations used to train Space Shuttle flight controllers and crew. This data can be used for testing from the lab either in real time or by playing back recorded data. The ONAV system also uses recorded, real data for testing.

Notice that connectivity to real-time data can introduce risk when using unconfigured software. The RTDS project resolved this difficulty by providing an independent data source which isolated prototypes from the operational data source. See also the discussion of technology integration into the support environment in section 6.3.

Storyboards

The storyboard technique employed for the PDRS HCI design concepts (Schreckenghost, 1990) proved useful as a means of identifying information requirements. Specifying these design concepts required early consideration of the operational use of the intelligent system. The storyboard served as a vehicle for discussing information requirements with operators that clarified misconceptions and improved domain understanding. It forced design of the workspace and provided examples of desired capability. A storyboard illustrating exemplary HCI designs in operational use represents a useful technique for integrating HCI into the design process of the intelligent system.

Problem: Providing Techniques to Identify Information Requirements

Recommendation: A storyboard of proposed HCI design concepts can be a useful tool for identifying information requirements. The storyboard can be used to illustrate simple operational scenarios, to refine the design concepts, and to elicit additional requirements.

It is recommended that each storyboard picture be supplemented by a text description stating the purpose and information content of the illustration to assist in identifying requirements. An index by topic is also recommended to allow easy access for review. Further investigation of methods for deriving requirements from storyboards is needed.

Issue: Techniques and tools are needed to assist designers in identifying information requirements from a storyboard of design concepts.

See figure 6-2 for an example of a requirement derived from one of the screens developed for the PDRS design concepts. This screen design was used as an illustration of the information required to perform domain tasks and does not represent a final user interface design. The designers task is to extract the information requirement's "message" embedded in the storyboard "medium".

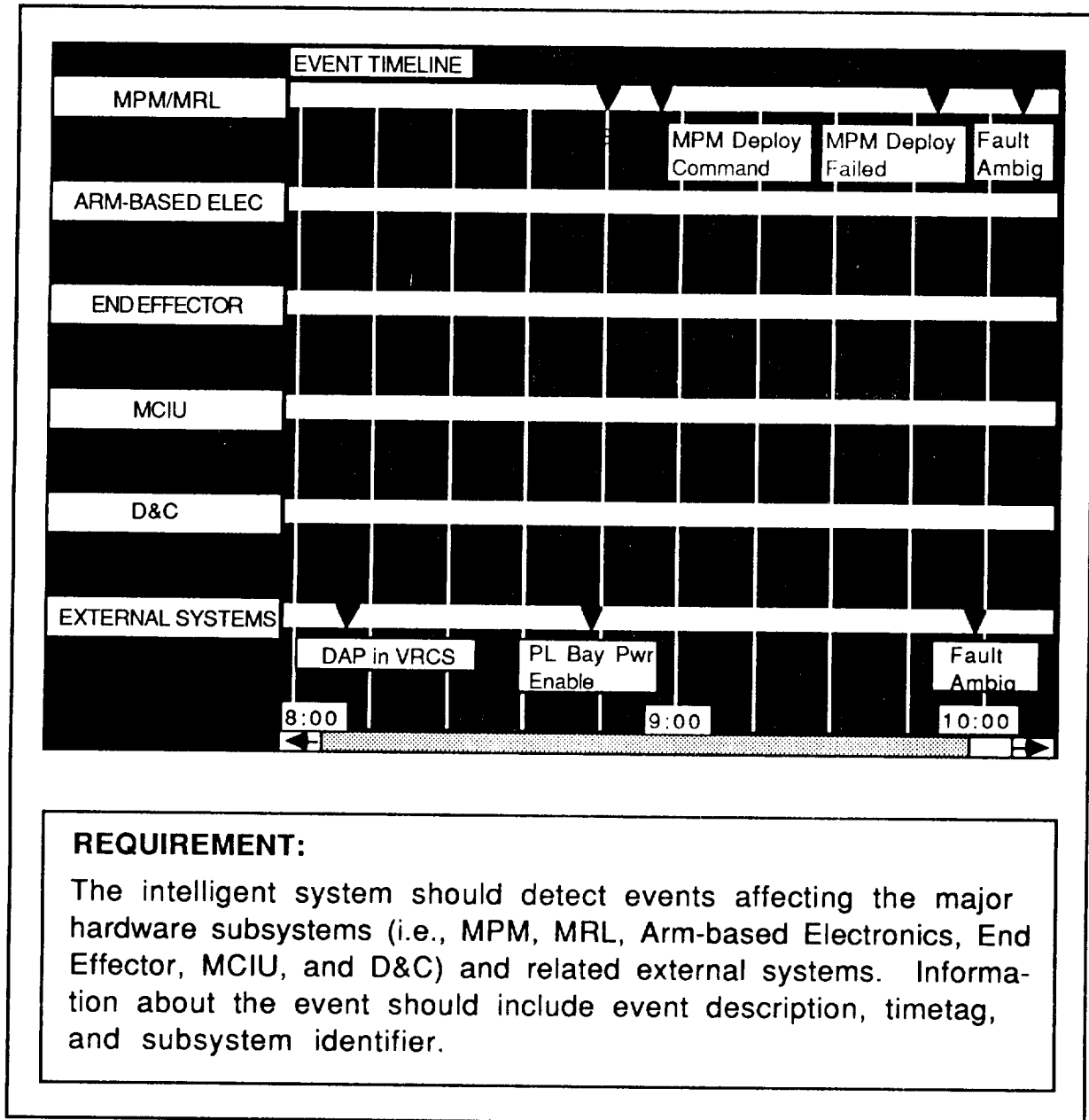


Figure 6-2. Example of Requirement Derived from Storyboard (PDRS HCI Design Concepts)

When demonstrating the PDRS HCI design concepts, it became obvious that the storyboard technique does not capture the reasoning behind design decisions. Frequent questions included "why did you do that?" or "what about doing something else?". A design methodology should

include specification of the constraints imposed on the design and the reasoning behind design decisions made to meet those constraints.

Problem: Documenting the Design Constraints

Recommendation: Constraints on the design should be clearly documented with the associated design decisions and justification. This includes constraints due to the operational environment, compromises due to the prototyping environment, and limitations imposed by the user community.

A number of important design constraints were observed in the case study. For example, an operational constraint for systems used in the Space Shuttle Mission Control Center is that no audio coding can be used. When using a HCI prototyping tool to implement the PDRS design concepts, a number of compromises from the original design were mandated by limited tool capability (e.g., difficult to use color, no black background). A limitation typically imposed by the Space Shuttle user community availability of the current user interface while transitioning to the new user interface provided by the intelligent system. For the ONAV system, this resulted in the requirement to emulate the existing display on the new display for testing.

6.1.3 Design Description

During iterative development, modifications can occur in both the requirements (e.g., a new information item is needed from the intelligent system) and the design specification (e.g., a display item should be moved on the screen). Fischer described this iterative development approach as "the coevolution of specifications and implementations" (Fischer, 1989). The levels of analysis for a computer-based display presented in Section 5.1.2 could provide a framework for specifying the design description, where the information requirements and the display specification are developed in parallel. In this model, design moves between levels as design constraints are identified and design decisions made. These levels of analysis will be investigated as part of the specification of requirements for a design methodology.

6.2 Design Team

The design methodology should support development by multi-member design teams. Such teams are typical of complex system development. Multi-member design teams can fragment into multiple, small design teams working on portions of the system. The design methodology should facilitate communication between developers that are working on different portions of the system. It should foster the concept of a single team with experts from multiple disciplines, including HCI expertise. An important aspect of integrating HCI design into system design is the participation of HCI experts on the system design team. The development methodology should also encourage user participation throughout the development process. Recommendations and issues affecting the design team are discussed in the following section.

6.2.1 Communication within Design Team

A prevalent problem in current approaches to system development is the lack of communication between designers responsible for the intelligent system and those responsible for the user interface. Often these design efforts are conducted in relative isolation with a minimum of information exchange about how the designs will work together. This isolation is frequently exacerbated by the delay of the user interface development until the end of the development

process. Such an approach prevents any consideration in the intelligent system design of information requirements resulting from the evaluation of HCI, since HCI is not investigated until it is too late to influence the intelligent system design.

An effective way to improve communication between designers and to integrate HCI into system design is to form a single, multi-disciplinary design team from the early stages of development. The concept of a "user interface design team" is replaced by HCI and user interface expertise in the system design team. A design effort involves multiple activities (e.g., software engineering, analysis of requirements, or visual design) often conducted in parallel. The design team should reflect the composite, multi-perspective nature of system design. It should include such skills as intelligent system development, software engineering, HCI expertise, user interface expertise, and domain expertise/user experience. HCI expertise would address such issues as effective agent (human and computer) architectures and information requirements for domain tasks and coordination of agents. User interface expertise would address such issues as identification and management of information overload, as well as graphic design, style of presentation, and application of user interface guidelines.

Problem: Selecting Members of the Design Team

Recommendation: The design team should be multi-disciplinary from the early stages of development, including expertise in such areas as intelligent system development, software engineering, human-computer interaction, user interfaces, the domain, and operations (i.e, users). HCI expertise should be distinguished from user interface expertise. HCI expertise addresses such issues as information requirements and coordination of agents. User interface expertise addresses issues of graphic design and style of presentation.

Mechanisms for communication between members of design team are being investigated by evaluation of the prototype toolkit User-Intelligent System Interface Toolkit (UISIT). This toolkit supports a design methodology which explicitly defines the information passing between the intelligent system and the operator (via the user interface) (Kessel, 1990). This information is captured as objects in a distinct layer of the system architecture called the communication layer. The communication layer also represents a common/global repository for information requirements useful to both intelligent system designers and user interface designers. The concept of a communication layer can be used to coordinate and couple intelligent system design with user interface design and provide a mechanism for designers to communicate their information needs. Information requirements in the form of an information object layer are being investigated as a means of communication and coordination.

Problem: Ensuring Communication between Design Team Members

Recommendation: The design methodology should provide mechanisms for communication and coordination between members of the multi-disciplinary design team. Information requirements seem a good candidate for such coordination.

The communication mechanism can assist in design of a more integrated application as well. Close coordination between team members is required to minimize redundancy and to ensure consistency, ease of navigation, and utility of display groupings. Design of the workspace should include a group effort to identify activities that span the designs of multiple individuals.

To achieve this level of integration, designers must make explicit much of the information usually implicit for a single designer. For example, tables describing designated colors or lists of available schematics can improve system consistency and minimize redundant effort.

Issue: When the user interface is designed by a multi-member design team, problems of consistency and usability can occur. Methods are needed to achieve an integrated workspace design with multi-member design teams.

6.2.2 User Participation in Design

Iterative development with incremental knowledge acquisition has resulted in active user involvement in portions of the development process. Such involvement has contributed to the success of these intelligent systems by easing technology transfer into the user community. The roles of the user during system development are varied.

The Space Station program plans to provide the capability for users to design their own displays (Wilford, 1990). A tool set called the User Support Environment (USE) will be provided for user definition of displays. An example of user-generated displays was observed in the case study. The RAVES application allows users to develop new display formats using a specific HCI tool, DataViews®. The completed display is specified as a visual screen design and a list of parameters presented in the display. This description is provided to system implementers for integration with the operational application (see Volume 2, Malin et al., 1991).

Typically, there are multiple users for a fault management application. For most of the cases surveyed, a subset of this user group was closely involved in design of the intelligent system. In some cases, such as the RTDS applications, the designers and implementers have included users. The remainder of the group serve as reviewers of the design.

A variety of user review techniques were observed. For the RTDS applications, prototypes were inserted into the operational environment for side-by-side test and review. Demonstration and review was another approach observed in the case study. An alternative technique is to build small, stand-alone prototypes for evaluating portions of the design. These prototypes allow users to perform hand-on testing of designs early in the development process. They are quickly modified based on user feedback, which facilitates rapid knowledge acquisition and refinement. The prototype can be used to evaluate the portions of the knowledge base, the HCI, or the user interface. The PDRS HCI design concepts were implemented using a prototyping tool for evaluation of the HCI.

Problem: Involving the Users in Design

Recommendation: Active user involvement in the design and development of the intelligent system and its user interface contributes significantly to the potential for success of the system. Frequent user review of the requirements and prototypes, including both demonstration and hands-on use, is recommended to improve both the accuracy of the design and user acceptance of the system. Feedback from these reviews should be incorporated into the system design.

® Data Views is a registered trademark of VI Corp.

An emphasis on user-generated software was observed during the case study as well as at advanced automation conferences (e.g., Space Station Evolution Symposium, February, 1990). Many of the users developing intelligent systems have minimal software development experience and no intelligent system development experience. For the RTDS applications, the need to assist users in developing intelligent systems influenced the selection of the tool G2, which provides a natural language syntax in its interface. Performance of complex systems and scalability of prototypes are other areas where users need assistance. Tools and methods to support software generation by users are needed.

Issue: Often, users are not trained in the development of intelligent systems or even in software engineering. This can result in problems such as software performance and scalability of prototypes. Tools and methods are needed to assist users in designing and implementing intelligent systems and their user interfaces.

6.3 Integration into the Support Environment

Typically, the intelligent system is one software element of a larger support system. Many elements of that system may be conventional, non-automated elements. Often, the support system contains software and hardware elements from a previous support system. The automated technologies must be united with the conventional technologies to form an integrated support system. The intelligent system designer should avoid designing the system as a stand-alone replacement for an existing approach to the task. The intelligent system design (including HCI and user interface design) should be embedded into design of the entire support system, including the design of operations.

Problem: Integrating Intelligent System Design into Design of Support System

Recommendation: Not only should HCI and user interface design be embedded in intelligent system design, but intelligent system design should be integrated into the design of the entire support system, including the design of operations.

Upgrades in an existing system can result in the inadvertent removal of easy-to-use tools and methods. For example, the Real-Time Interactive Monitoring System (Volume 2, Malin et al., 1991) failed to provide a hardcopy capability with its strip chart emulation. A typical procedure during post flight analysis was the review of these hardcopies, so this loss of capability was an impact to users. Norman (1990) discussed methods for manipulating physical checklists that can be lost with electronic checklists (e.g., operator can hold the checklist such that his thumb points to current location within checklist).

Changes in human-computer interaction resulting from such upgrades can also impact the operator's ability to gain or maintain expertise. The operator can be incidentally informed about specific processes and systems while using the user interface to perform nominal tasks (Bloom, 1991). For example, monitoring raw data for anomalies also informs the operator of a model of the data (e.g, characteristics such as range of values, rate of update, behavior trends, frequency and duration of transients). This training can be inadvertently lost when using a symbolic interpretation of the data. The new design should provide equivalent opportunities to learn on the job and should reinforce such acquisition of expertise. See also section 4.1.1 for a discussion of task allocation that includes on-the-job training with real-time support.

Problem: Avoiding Negative Impacts Caused by System Upgrade

Recommendation: Upgrades to an existing system can incidentally complicate the operator's job or negatively impact his ability to maintain job expertise. The designer should carefully evaluate current procedures (documented and undocumented) before removing or altering existing methods and tools. He should also identify opportunities for the operator to learn on the job and incorporate these in the new design.

A cautious approach should be taken when integrating a new or different technology into an existing environment. The availability of new information from that technology can represent increased risk in operations. If the system implementing the technology is not designed properly and integrated with existing tools, the workload of the operator can be increased by providing yet more information to interpret. Incidental loss of information can also occur, impacting the operator's ability to maintain situational awareness (see the discussion in section 4.2.2 on visibility into the monitored process). Total reliance on a new software application should be avoided. For example, Leveson (1991) discussed a situation where a software failure in new medical radiation equipment resulted in lethal dosages of radiation during treatment. The accident was possible because the hardware interlocks that would have prevented administering lethal amounts of radiation had been removed when the new software capability was added. Instead of precipitously removing old capability, there should be a gradual integration of the new with the old. Adequate information to diagnose and compensate for failures should also be provided. Recommendations for managing software failures in the intelligent system are discussed in section 4.1.3.

The constraint to retain portions of an existing system in parallel with the upgraded application represents the gradual integration of the new technology into an existing environment. A gradual approach to technology insertion was evident throughout the case study applications. By far, the most common development approach was multiple, iterative cycles of testing, evaluation, and modification (Volume 2, Malin et al., 1991). Each cycle consisted of small upgrades that were quickly implemented and tested (often in an environment similar to the operational environment). When an upgrade represented to large a technical leap, the user would often resist the design and the designer would be required to redesign with less capability (e.g., the original INCO display was considered too advanced). This incremental approach improved operator trust of the new technology by providing the user frequent opportunities for feedback and allowing gradual accommodation to change (Gnabasi, 1990).

Problem: Integrating a New or Different Technology into an Existing Environment

Recommendation: The integration of a new technology into an existing operational environment should be approached gradually. An iterative development process with frequent small upgrades from the current system is an effective way to introduce the technology. Each upgrade should be followed by a period of operator review and use in an environment similar to the operational environment.

An example of the gradual integration of new technology with old has occurred in the Space Shuttle Mission Control Center (MCC). As a part of the upgrade to the MCC, workstations have recently become available for flight control support. The Real Time Data System (RTDS) was created to provide access to telemetry data from these workstations via an independent data source. This approach has allowed the operators to prototype new technologies, such as

advanced user interfaces and intelligent systems, and test them in an operational environment without introducing risk due to unreliable, unconfigured software. See section 6.1.2 for a discussion of evaluation of prototypes and Volume 2 (Malin et al., 1991) for a description of RTDS.

There is also the risk of building an inflexible capability when integrating a new technology with an older, existing technology. The existing technology may not be able to fully support the new technology at the time of integration. For example, information currently in paper documents (e.g., Space Shuttle procedures, mission timelines), software source code (e.g., rules) and on the voice loops will become available in an electronic format eventually. The new system should include hooks for such expected evolution, including planned improvements to the existing environment.

Problem: Considering System Evolution during Design

Recommendation: The intelligent system and its user interface should be designed to ease evolution to planned improvements to the existing environment. Hooks should include designing for transition to alternate sources of information or automation of currently manual functions.

For example, in the PDRS HCI design concepts, a pop-up window is provided for access to information about existing fault ambiguities. Included in this information are references to the location in a manual of the procedures that could reduce ambiguity and a reference to related flight rules. If properly implemented, the pointer to a list of references could be easily replaced with a pointer to the electronic version of the listing of the procedure or flight rule at a later time.

Tools and methods used to perform current operations can be a source of useful design information when integrating new HCI technology into an existing environment. Consideration of current operations promotes consistency and familiarity and improves operator acceptance. There is a trade-off, however, between designing a system for integration with an existing system and avoiding retention of bad designs by rigidly following the old way of doing business, which may be an artifact of out-dated technology (e.g., unavailable graphics results in use of paper schematics only) (Brooks, 1991). This trade-off should be focused on identifying the user's goals and supporting the user's decision-making process (e.g., information should be represented consistently with the planned use of the information). For example, if the task requires that two values be compared, the intelligent system should be designed to make certain that both values are available when needed and the user interface should be designed to allow easy comparison.

Problem: Matching the System Design to the User's Task

Recommendation: When adding new technology to an existing operational environment, it can be very useful to look at manual techniques and off-line devices for ideas to incorporate into the design. Existing techniques and methods of information presentation should only be adopted, however, when they are consistent with the planned use of information in support of the user's decision-making process.

For example, the user interface to the GNC Jet Control application (Volume 2, Malin et al., 1991) directly incorporated a diagram of the jet locations and thrust directions from the paper documentation. This paper diagram was used to predict the impact of the loss of specific jets. The intelligent system incorporated this diagram and provided interaction capability to emulate the manual technique that had always been used by the flight controllers.

Another example of bringing off-line capabilities on-line was seen in the applications built for PDRS flight support. The Position Monitor is a graphical display of the three dimensional projection of the Space Shuttle, the arm, and the payload. This display was originally developed for use with an off-line simulation. The Position Monitor adopted the same display format, but connected it to downlisted telemetry, so it reflects actual position instead of simulated position.

6.4 Delivery of Design Guidance

In sections 4 and 5, recommendations to assist designers in developing intelligent systems with effective HCI were provided. These recommendations focused on the identification of information requirements for the intelligent system and its user interface. Merely identifying design guidance is not sufficient, however. Use of HCI design guidance must become a part of the intelligent system development process if it is to be effective. One of the purposes for evaluating the design process was to provide insight on mechanisms for integrating design guidance within this process. HCI design guidance should assist the designer at all stages of the development process (see section 6.1 for a description of these stages). Methods and tools are needed that integrate the use of HCI guidance into the system design process. The development methodology should not only permit the use of design guidance, but should make guidance easier to use and incorporate use of guidance as an essential element in design. The integration of design guidance into development methodology includes both traditional forms of user interface guidance as well as the information-level guidance presented in this report.

Issue: The development methodology should make guidance easier to use and should incorporate use of design guidance as an essential element in design. Methods and tools are needed that integrate the use of HCI guidance into the system design process.

A likely approach to facilitating use of design guidance is computer-based delivery as part of an integrated design methodology. Computer-based delivery is an useful alternative to paper-based delivery. Although documents are easily disseminated into multiple environments, use of documents can be frustrating and time-consuming. A well-recognized problem with paper delivery of guidance is difficulty in accessing relevant guidance. The categorization of guidance for easier access can hide relationships that are not represented by the selected categories. Also, information access in a paper-based format is mandatorily sequential in nature. Methods are needed to provide multiple access points to design guidance and to assist the designer in identifying useful relationships between recommendations and examples. A prime candidate for investigation is hypermedia access.

A preliminary format for the delivery of design guidance has been proposed. See figures 6-3 and 6-4 for an illustration of HCI design guidance. For each design recommendation, the following information would be provided:

- Topic
Keywords that describe the issue area.
- Problem
A statement of the design issue or difficulty.

- **Recommendation**
A brief statement of the recommended design guidance.
- **Supporting Information**
The foundation or evaluation supporting the recommendation. This can include alternatives to and impacts of the guidance.
- **Example Illustrating Problem**
An example illustrating the difficulty BEFORE use of the recommendation.
- **Example Illustrating Recommendation**
An example illustrating the improved design AFTER use of the recommendation.
- **Techniques**
Suggested methods for implementing the recommendation. These may address information presentation as well as information content.
- **Cross References**
Identification of other sections containing related design guidance.
- **Research Issues**
Related issues requiring further investigation.

EXAMPLE OF HCI DESIGN GUIDANCE

Topic: Intervention Into Intelligent System Reasoning Process

Problem:
An intelligent system can become "disoriented" and require the operator to redirect it onto a more productive path of investigation (e.g., the system investigating an unproductive hypothesis or failing to investigate a likely hypothesis).

Recommendation:
Operator intervention into the reasoning process of the intelligent system can be used to manage errors in the intelligent system. Methods of intervention include modification of the reasoning process or selection of an alternate reasoning mechanism.

Supporting Information:
Intervention into intelligent system processing is one approach for redirecting a "disoriented" intelligent system onto a more productive path of investigation. The intelligent system must be designed, however, to permit such intervention. To effectively intervene in processing, the operator must first have a good understanding of the reasoning strategy used by the intelligent system.

Example Illustrating Problem:
See the following page for an example that illustrates the problem.

Example Illustrating Recommendation:
See the following page for an example that illustrates the recommendation.

Techniques:
Methods of intervening in the method of processing information include modification of the reasoning process or selection of an alternate reasoning mechanism. Examples of modification of the reasoning process are (1) setting processing priorities (e.g., what hypothesis to investigate first), (2) alteration of hypotheses, and (3) specification of alternate solutions.

Cross References:
The user interface used to intervene should reinforce the operator's understanding of the reasoning process to assist in identifying appropriate action. See section 4.1.2 for a discussion of presenting information about intelligent system reasoning. See the discussion of risk in intervening in the intelligent system later in this section.

Research Issues:
Making intelligent systems easily interruptable and redirectable, developing vocabularies for communicating advice about control decisions which reduce the amount of knowledge required by human team members about intelligent system internals.

Figure 6-3. Example of HCI Design Guidance, Page 1

EXAMPLE OF HCI DESIGN GUIDANCE

Topic: Intervention into Intelligent System Reasoning

AGENT INTERACTION ILLUSTRATING PROBLEM

In this case, the operator is unable to communicate information about a likely fault and its correction procedure to the intelligent system. Unaware of this information, the intelligent system would continue to pursue fault isolation on an incorrect set of suspected faults, even when the actual fault was corrected. The operator would have to do a restart to reset the knowledge base and "fix" the intelligent system. Figure illustrates the agent interaction.

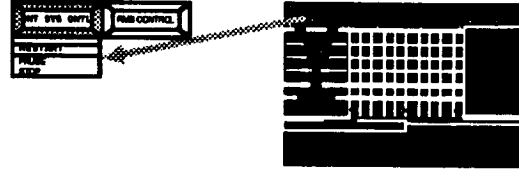


Illustration: Operator Restarting Intelligent System

AGENT INTERACTION ILLUSTRATING RECOMMENDATION

In this case, the ability to affect reasoning by altering hypotheses used by the intelligent system has been provided to the operator. The operator "informs" the intelligent system about the fault, performs a corrective procedure, and normal processing continues. Figure illustrates the agent interaction.

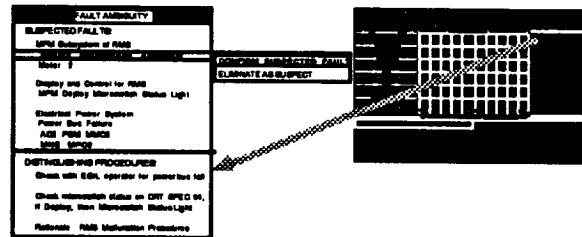


Illustration: Operator Informing Intelligent System of a Fault

Figure 6-4. Example of HCI Design Guidance, Page 2

Section 7 Summary

A variety of difficulties can be encountered when introducing an intelligent system into fault management operations. Adverse changes in the availability of information and the assignment of tasks can result. Intelligent systems represent a new source of information that can contribute to the existing problem of information overload in real-time flight support. Inappropriate task assignments can complicate task performance or distract the operator during critical operations, inadvertently resulting in performance degraded from that of the operator alone. Additionally, changes in information and tasking can result in lost opportunities for on-the-job training and development of expertise.

A second difficulty arises because the intelligent system represents another source of error. Without control of the intelligent system, the operator can lose the ability to make task decisions. Control of the intelligent system requires the capability to intervene in intelligent system processing to inform and redirect the system. Additionally, the levels of automation provided by the intelligent system are often misunderstood or ignored, which affects operator control of the intelligent system, leaving the operator uncertain of task responsibilities or even unaware of available modes of operation.

These difficulties arise from failing to consider HCI in the design of the intelligent system and its user interfaces. Traditional forms of user interface design guidance emphasize the medium (i.e., how information is presented) and not the message (i.e., information content). Often what is perceived as a user interface design problem is actually an intelligent system design problem (e.g., unavailable information). There is a need for assistance in defining the information exchanged between human and computer (i.e., information requirements).

Design Guidance

Many of the insights into design guidance result from considering the user and intelligent system as members of a fault management team. A new perspective of the user results -- the user as another type of agent in a heterogeneous, cooperating, distributed system. System design then becomes the design of an architecture for accomplishing domain tasks with the available human (i.e., users) and computer (i.e., intelligent system) agents. This architecture must support multi-tasking, dynamic task assignment, and shared agent tasking.

The close agent interaction that results from the team concept requires that human and intelligent system collaborate to accomplish tasks. Common forms of explanation, retrospective in nature and based on static rationale or simple rule traces, are inadequate. Collaboration requires that the user have visibility into the reasoning behind intelligent system conclusions and recommendations. In effect, the user must be able to share the world view of the intelligent system. The intelligent system must be designed to support this type of collaboration.

To be an effective team member, the intelligent system must be directable. This includes possessing the flexibility to allow real-time reallocation of tasks as well as providing capability for the operator to inform and redirect the intelligent system. Tasks may be reallocated when anomalies occur or mission goals are altered or to compensate for an overworked agent. The ability to inform the intelligent system of new or revised information can be used when data are not available in electronic form (e.g., information on voice loop) or when the intelligent system has been misinformed (e.g., noisy or erroneous data). Redirection can be used to compensate for intelligent system errors, improve system performance, and set the intelligent system on a more productive path of investigation.

There are a number of information requirements for supporting fault management of the monitored process. The intelligent system should assist the operator in interpreting alarms and minimizing false or redundant alarms. Critical diagnostic information should be accessible, improving visibility into the monitored process and supporting the operator in evaluating the consequences of events and planned activities, monitoring and executing procedures, and assessing functional capability after a failure. In the event of an unanticipated situation, the team should be able to generate and execute workaround procedures.

One of the common problems encountered in real-time support systems is overloading the operator with information, especially when an anomaly occurs. Anomalous conditions are often accompanied by warnings, alarms, new events, and activity changes. Managing such information overload requires identifying the important message contained in this flood of facts, i.e., interpreting the information. Methods that improve the operator's ability to interpret information include use of information context in presentation, qualitative representation of information consistent with the operator's mental models, and techniques for summarizing information and reducing irrelevant detail.

Several trends in user interface design were found based on the cases studied. One issue that occurred in several systems is a proliferation of windows which can lead to navigation issues concerned with where to find related data. This is complicated by hidden windows and complex menus. This trend points to a lack of workspace design (coordinating the set of views into the monitored process as well as the intelligent system that can be seen together in parallel or in series) and lack of specification of information requirements (what information should the observer be able to extract from observing a particular display -- in isolation and in concert with other displays). With respect to the interface between the human operator and intelligent system, diagnostic reasoning was typically displayed as chronologically ordered message lists. However, this approach fails to capture the temporal nature of events or distinguish between kinds of messages (e.g., events and actions). The predominant view into the monitored process was physical topology schematics annotated with active data about the state of the monitored process (i.e., color coded digital parameter values or component states). However, there were cases in which this approach did not provide appropriate information for the operator (e.g., did not highlight events or system change).

The net result of these user interface trends is black-box, inscrutable systems where the user interface capabilities inadvertently reinforce barriers between the human operator and the intelligent system as well as between the operator and the monitored process. The lack of transparency in inscrutable systems, caused by lack of needed information or confusing presentation of excessive information, makes it difficult for the operator to visualize the intelligent system's situation assessment and recommendations in relation to the flow of events in the monitored process.

Design Methodology

The availability of HCI design guidance alone is not sufficient. Design guidance must be used to be effective. Thus, it is also important to identify how to integrate HCI design into design of the support system (both conventional and intelligent portions). A design methodology is needed that makes guidance easier to use and integral to the development process. The evaluation of system design processes observed in the case study has yielded the following goals for an integrated design methodology:

- Supporting the development of a task definition at the level of actions for domain tasks and agent coordination, not user interface actions

- Assisting designers in extracting requirements and design specifications from prototypes and storyboards, for both the intelligent system and the user interface
- Supporting the use of operational scenarios for developing and evaluating design concepts that consider HCI issues
- Defining HCI guidelines (both for identifying information requirements and specifying the user interface design) and making use of HCI guidance an integral part of intelligent system development
- Providing mechanisms for coordination and communication within the diverse membership of development teams typical for complex systems

The development methodology should map the task definition into requirements for information essential to effective HCI. These information-level requirements impact all elements of the system (i.e., application, user interface, user). A design methodology for developing information requirements should include the following steps: (1) description of domain tasks in terms of goals and actions required to achieve goals, (2) identification of resources provided to perform tasks and the constraints that affect task performance, (3) specification of agent activities and valid agent behaviors in an architecture for multi-tasking and dynamic task allocation, (4) evaluation of the activity specification using complex activity sequences and modes of agent interaction, (5) analysis of requirements for system information, (6) design of application and user interface using information requirements.

Consistent with the goal of integrating HCI design into design of the support system, it is necessary to form a single system development team. This team should be multi-disciplinary from the early stages of development. It should include such diverse personnel as software designers, users, HCI experts, and user interface experts. Notice the distinction between HCI expertise and user interface expertise. HCI experts assist with issues of agent coordination and in the definition of information requirements. User interface experts assist with issues of graphic design and style of presentation. User participation throughout system development is essential. The design team should adopt a development methodology that supports communication between members of the design team. Information requirements, represented as objects in a distinct layer of the system architecture, are currently under investigation as a possible communication mechanism.

Design Issues

Much work remains to be done to realize the goal of providing effective HCI design guidance for intelligent systems. Team architectures must be defined that specify how fault management tasks are performed, including agent interaction, mode shifts, and dynamic task assignment. This investigation includes identifying what coordination activities are required for shared tasking and information requirements for a controllable, directable intelligent system.

A second issue area concerns identifying information requirements that support agent communication and collaboration. These requirements must provide visibility into intelligent system reasoning and permit the operator to understand how and why the intelligent system reached a conclusion. Methods to support collaboration that go beyond retrospective explanation are required. These methods should be based on creating a shared view of the world between agents, including shared goals. This shared view is essential for both agent collaboration and effective intervention into and control of the intelligent system.

A third issue area concerns design of user interfaces to accommodate the new types of information and graphic forms from intelligent systems for real-time fault management. New display designs are needed to handle these new types, to handle the new mix of tasks and information, and to support management of the intelligent system.

Finally, an intelligent system development methodology is needed to facilitate use of HCI design guidance in developing information requirements and in designing intelligent systems and their user interfaces. Areas of investigation for development methodology include (1) modified techniques for task analysis to support definition of agent coordination activities, (2) methods for using operational scenarios to evaluate prototypes and storyboards, (3) methods for extracting and representing information requirements from prototypes and storyboards (4) use of information requirements as the basis for communication between members of design team.

Meanwhile, we hope that this document provides significant assistance to designers of intelligent systems for real-time fault management . We also hope that it can be used by the research communities of artificial intelligence, human factors, and software engineering to identify issues to investigate so that even better assistance can be provided to intelligent system designers in the future.

Appendix A

Points of Contact for the Case Study

Space Shuttle Real-Time Data System

Johnson Space Center
Troy Heindel/NASA

Space Shuttle Guidance, Navigation & Control (GNC) Intelligent Systems

Johnson Space Center
Dave Miller/RSOC
Ron Montgomery/RSOC

Space Shuttle Instrumentation and Communications Officer (INCO) Expert System Project

Johnson Space Center
Art Rasmussen/MITRE

Space Shuttle KU Band Self Test Expert System

Johnson Space Center
George Pohle/RSOC

Space Shuttle DATA COMM Expert System

Johnson Space Center
George Pohle/RSOC

Space Shuttle Payload Deployment and Retrieval System Decision Support System (DESSY)

Johnson Space Center
Don Culp/RSOC
Dave Mayer/RSOC
Joe Watters/formerly RSOC
Kristen Farry/formerly RSOC
Gordon Johns/MITRE
Mark Gnabasik/MITRE
Mary Czerwinski/ formerly LESC
Benjamin Beberness/LESC

X-29 Remotely Augmented Vehicle Expert Systems (RAVES)

Ames Research Center Dryden Flight Research Facility
Dale Mackall
Dorothea Cohen

Military Aircraft Real-time Interactive Monitoring Systems (RTIMES)

Edwards Air Force Base
Robin Madison

Space Shuttle Onboard Navigation (ONAV) expert system

Johnson Space Center
Lui Wang/NASA
Malise Haynes/NASA

Space Shuttle Rendezvous Expert System

Johnson Space Center
Hal Hiers/NASA
Oscar Olszewski/LESC

Space Station Procedures Onboard Management System (OMS) Prototypes
Johnson Space Center
Christine Kelly/MITRE
C.Jayne Guyse/MITRE
Dave Hammen/MITRE
Chris Marsh/MITRE

Space Station Module/Power Management and Distribution (SSM/PMAD)
Marshall Space Flight Center (MSFC)
Dave Weeks/NASA
Bryan Walls/NASA

Space Shuttle Knowledge-Based Autonomous Test Engineer (KATE)
John F. Kennedy Space Center (KSC)
Jack Galliher/NASA
Carrie Belton/NASA
Barbara Brown/NASA
Steve Beltz/Boeing

Space Shuttle Intelligent Launch Decision Support System (ILDSS)
John F. Kennedy Space Center
Arthur Beller/NASA
H. Greg Hadaller/Boeing

Appendix B

Disturbance Management

The main target application for this work is fault management where there is some monitored process (an engineered system) whose state changes over time. Faults disturb the process and diagnosis goes on in parallel with manual and automatic responses to maintain process integrity. In fault management the action is swift, the consequences high, and the situation saturated with uncertain data.

The operator may need to satisfy multiple competitive goals in the face of incomplete and often contradictory information. There is high pressure to perform efficiently (the need to get through a certain number of operations each day, or to land a certain number of aircraft per hour), and omnipresent in the background is the fact that the mission may fail or the plane may crash. The expert in these worlds is often confronted with resource saturation, especially at high criticality time periods. The strategies that experts use to cope with these demands is particularly relevant to those who would design information systems to support their activities.

Expert performance is more than simply following a plan or collection of guidelines. Rather the expert is one who can adapt plans, bring new plans into being and cancel others as new events warrant, and maintain several threads of action in different stages of completion. The critical contribution of people to the person-machine ensemble is adaptability in the face of the variability and surprise of real, complex situations (Rasmussen, 1986; Woods, 1988). This adaptability means that experts can handle special cases and exceptions as well as routine cases; to use efficient reasoning shortcuts but then to switch to more thorough reasoning strategies when cues indicate that the present case is atypical.

The human operator must track evolving situations loaded with unanticipated and potentially threatening events. As a result, operators must build and maintain a coherent situation assessment in a changing environment where multiple factors are at work including one or more faults, operator interventions and automatic system responses. How do people evaluate large amounts of potentially relevant and changing data in order to size up a situation in the face of time pressure? Researchers who examine expertise in situ have noted that practitioners themselves coin various phrases that describe the ability to maintain this coherent view of changing situation: in commercial aviation it is referred to as being "ahead of the plane", in carrier flight operations the expression "having the bubble" is used (Roberts and Rousseau, 1989), in military operations von Clausewitz called it "coup d'oeil" -- the ability to discern where and when a decisive action can be taken.

Attentional control in the face of multiple interleaved activities and the possibility of asynchronous and unplanned events is a fundamental part of fault management. Experts need to be able to manage several threads of activity in parallel, devoting enough attentional resource at the appropriate time in order to keep each on track. Also at issue here is interrupt handling -- as data changes and new events are noted, how do they or should they modify the current task or cognitive resource priorities. Understanding action in the face of diverse, changing and highly uncertain situations depends critically on understanding attentional processes and the dynamic prioritization of tasks. A critical criterion for the design of the fault management systems is how they support operator attention focusing, attention switching and dynamic prioritization.

B.1 Cascade of Disturbances and Disturbance Management Cognitive Task

Fault management in dynamic applications has a different character than the stereotype about diagnostic situations which is based on the exemplar of troubleshooting a broken device which has been removed from service. In dynamic process applications, fault management incidents

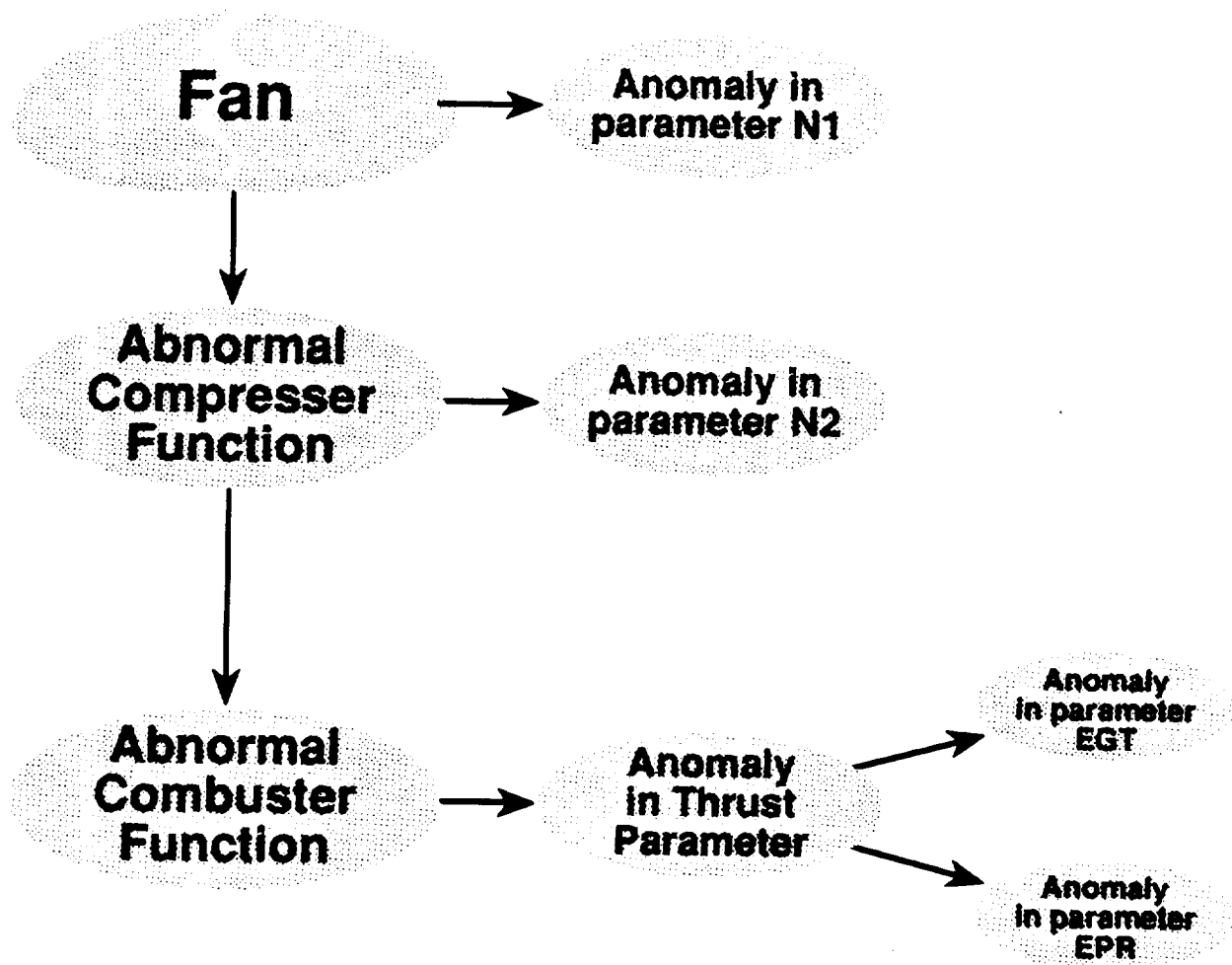
extend, develop and change over time. A fault disturbs the monitored process and triggers influences that produce a time dependent set of disturbances (i.e., abnormal conditions where actual process state deviates from the desired function for the relevant operating context). This cascade of disturbances unfolds over time due to the development of the fault itself (a leak growing into a break) and due to functional and physical interconnections within the monitored process (Woods, 1988; Abbott, 1988).

Figure B-1 provides an aviation illustration of the cascade of disturbances that can follow from a fault (adapted from Abbott, 1988). The initiating fault is a failure in the fan subsystem of an aircraft engine. This fault directly produces an anomaly in one engine parameter, but the fault also disturbs compressor function which is reflected symptomatically in an anomaly in another engine parameter. The effect of the fault continues to propagate from the compressor to the combustor producing anomalies in two more engine parameters. Diagnosis involves understanding the temporal dynamics of the cascade of disturbances. For example in this case, the temporal progression is an important clue to understand that the fault is in the fan subsystem and not in the compressor or the combustor. Note that, because of disturbance propagation, the same or a similar set of anomalies may eventually result from a fault in a different subsystem. A critical discriminating difference is the propagation path as the cascade of disturbances develops over time.

In dynamic fault management, the monitored process is not and usually cannot be removed from service. This means that the fault manager needs to try to continue to meet the goals of the monitored process (Woods, 1988). The relative importance of different process goals may change as the incident evolves and some goals may need to be abandoned if they compete with more critical goals (mission control activities following the oxygen tank explosion during Apollo 13 are a good example of this).

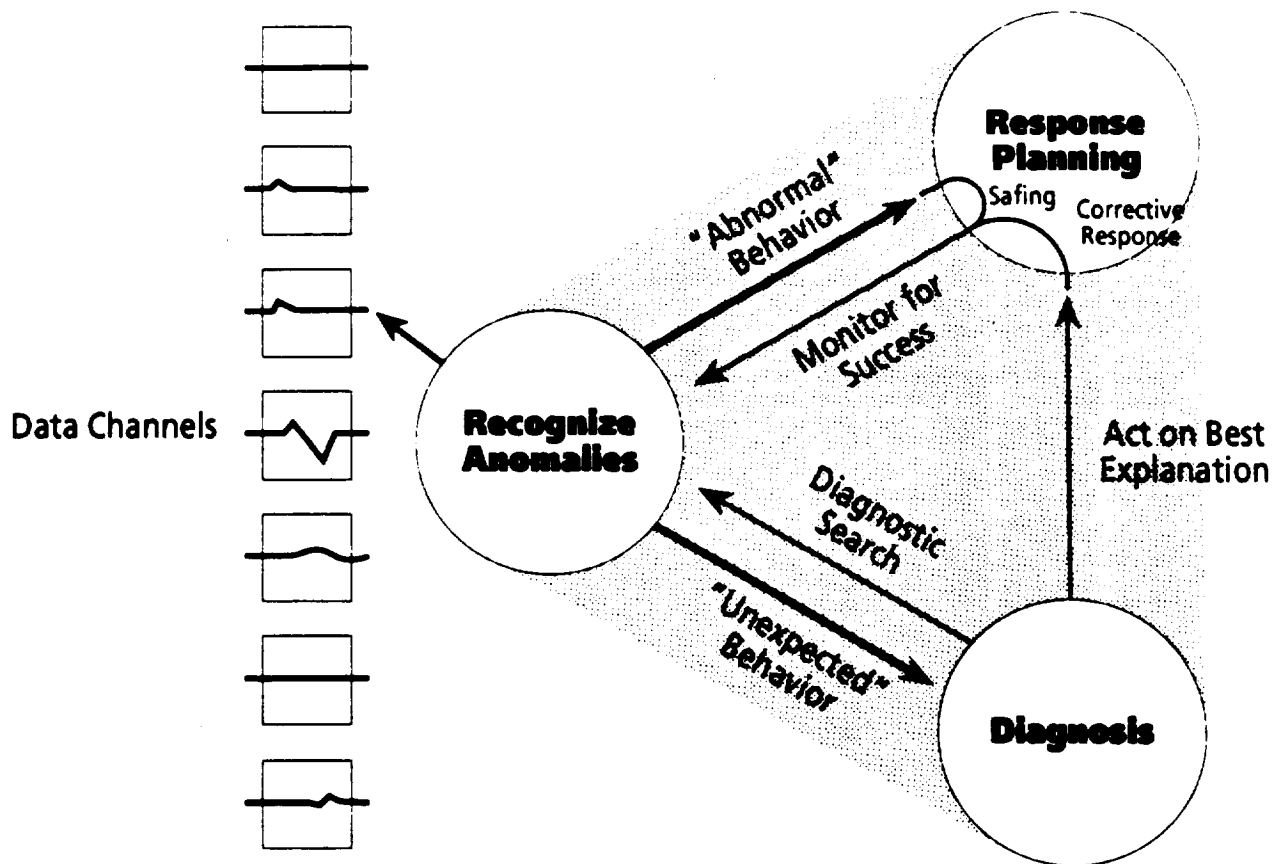
Thus, in dynamic, uncertain, and dangerous domains, fault diagnosis occurs as part of a larger context where the expert practitioner must maintain system integrity by coping with the consequences of faults (i.e., disturbances) through safing responses in parallel with untangling the causal chain that underlies these disturbances in order to take corrective responses. The interaction between these two lines of reasoning and activity defines a major cognitive activity of human experts in dynamic problem solving situations, what Woods (1988) has called the disturbance management cognitive task (figure B-2).

Information processing in fault management is anomaly driven. There are a large number of data channels and the indications on these channels may be changing (the left side of figure B-2). The first task of a fault management system (either human alone, machine alone or the ensemble) is to recognize, out of all the signal states and changes, which represent anomalies -- significant findings about the current and future state of the monitored process. Obviously, this can be relatively easy when all data channels are quiescent except for one. But faults in the monitored process produce multiple effects that change over time creating the potential for an avalanche of changing indications. The task for a fault management system is to recognize, out of all of the changing indications, which represent anomalies. Note that this is an example of a potential data overload situation where the critical cognitive activity is filtering the relevant indications from the irrelevant variations in the disturbed process (Woods, 1991).



© 1991 Woods, Potter, Johannesen, Holloway

Figure B-1. Aviation Example of Cascade of Disturbances that can Follow from a Fault
(adapted from Abbott, 1988)



© 1991 Woods, Potter, Johannesen, Holloway

Figure B-2. Model of Anomaly-driven Information Processing in Disturbance Management

This means that a critical characteristic of a fault management system from a cognitive point of view, is how it helps segregate the relevant variations from the irrelevant ones. And the critical constraint on carrying out this cognitive function is the context sensitivity problem -- which variations are important depends on the state of the process itself and on the state of the problem solving process. We can term this the alarm handling function of a fault management system.

There are classic paths that have been used to cope with the alarm handling demands of fault management. One is to develop a fixed, static priority assignment to individual alarm signals. Usually, two or three classes of priority are defined and then individual alarm signals are assigned to one these categories. Presumably, there are only a few high priority alarms that occur in the same time period and alarms in the lower priority classes do not need to be processed in order to evaluate the significance of the high priority ones. In other words, the static priority technique tries to cope with alarm handling demands through a scale reduction process.

Another classic technique is to develop automated fault diagnosis. The need to handle alarm information is avoided by simply developing a machine to do the diagnosis via heuristic or algorithmic computer processing. The automated diagnostic system processes the alarm information and determines what fault or perhaps what faults are present in the monitored process. When the system has determined a fault, the human operator is notified of the result. Now, the fault determination is often softened (in part because of reliability concerns) and output with an attached "degree of belief" marker, as a ranked list of hypotheses, or as a recommendation. Nevertheless, all of these approaches attempt to cope with the alarm handling demands of fault management through a finesse of allocating the task to a machine rather than supporting the human operator.

We mentioned that fault management information processing is anomaly-driven. In everyday usage, an anomaly is some kind of deviation from the common order or an exceptional condition. In other words, an anomaly represents a mismatch between actual state and some standard. To characterize a fault management system cognitively, one must specify the different kinds of anomalies that the system can recognize and information processing that is needed to recognize these classes of events.

One kind of anomaly has to do with departures from desired system function for a given context, i.e., the monitored process is not performing the way it was designed to perform. It could be that pressure is supposed to be within a certain range but that it is currently too low. Let us call this class of anomalies "abnormalities," that is, observed monitored process behavior is abnormal with respect to the desired system function for a particular context (e.g., shutdown versus full power operations).

Another kind of anomaly has to do with process behavior that deviates from the operator's model of the situation. In this case process behavior deviates from someone (the operator's) or something's (the intelligent system's) expectations about how the process will behave. The agent's expectations are derived from some model of the state of the monitored process. Because we are focusing on dynamic processes, this model refers to the influences acting on the process -- influences resulting from manual actions, influences resulting from automatic system activities, or influences resulting from the effects of faults. Anomalous process behavior that falls into this class we can call "unexpected," that is, observed monitored process behavior is unexpected with respect to model derived expectations for the particular context.

For example, if you trip a power generation system and there is some kind of cooling reservoir in the system, then level in that cooling reservoir is going to drop. It always drops when you shut off the power generation system. Thus, a low level alarm indicates an abnormality with

respect to the desired system function; however, the alarm is expected given the circumstances. The operator knows "why" the alarm indication is present (it is an expected consequence of the influence of the rapid shutdown) and therefore this alarm does not interrupt or change his or her information processing activities, for example, the operator will not try to "diagnose" the fault. What would be unexpected would be the absence of this alarm or if the low level condition persisted longer than is expected given the influence of the trip event. Note that there can be other kinds of anomalies as well, for example, departures from plans.

Figure B-2 illustrates anomaly driven information processing. Recognition of "abnormal" process behavior should lead to information processing about how to cope with the indicated disturbance, for example, safing responses. This, in turn, leads to monitoring lines of reasoning -- checking to see if coping responses have occurred as expected and whether they are having the desired effect. Thus, in the above example, the low level alarm should trigger a line of reasoning to evaluate what coping responses should be initiated to deal with the abnormality, for example, an automatic makeup system should start up to resupply the reservoir and a line of reasoning to monitor that the automatic system came on properly and is restoring level to the desired range. Recognition of "unexpected" process behavior should lead to diagnostic information processing -- a line of reasoning to generate possible explanations or "diagnoses" for the observed anomaly and knowledge-driven search to evaluate the adequacy of those possible explanations. When a diagnosis is reached (a best explanation), it can trigger a line of reasoning to identify or develop corrective responses.

This model of the cognitive activities in fault management has several implications for the design of intelligent systems to support fault management. One is that the fault management support system should help the operator see anomalies in the monitored process. Since anomalies are defined as mismatches, the fault management support system should help the operator see what specific mismatch is present. Since there are different kinds of standards for process behavior, e.g., target values, limit values, automatic system response thresholds, intelligent system "expectations" (in the case of model based AI systems), indications of an anomaly should include the standard violated.

Cognitive activities in fault management involve tracking the set of anomalies present in the process and their temporal inter-relationships. Fault management support systems can help operators see the dynamics of anomalies and the underlying disturbances in process functions, especially to see how disturbances grow and subside in the face of safing/corrective responses (Woods et al., 1986). This information may be very important in the diagnostic process and in the strategic allocation of cognitive resources either to diagnostic search to identify the source of the cascade of disturbances or to focus on coping/safing actions to protect important goals.

A fundamental feature of the disturbance management cognitive task is that diagnostic activities and information are intermingled with manual and automatic responses to cope with the consequences of faults. How the monitored process responds to these coping/safing actions provides information for the diagnostic process. In fact, people will often take actions whose primary purpose is to check out or confirm a hypothesis about the source of the trouble -- diagnostic interventions. It is important for a fault management support system to assist the human operator untangle the interaction between the influences of fault(s) and the influences of coping/safing actions taken by automatic systems or by some of the people involved.

B.1.1 Alarm Handling Trends in the Case Study

With respect to fault management and with the very notable exception of the Selective Monitoring system (SELMON), most of the systems surveyed relied on the automated

diagnosis technique.¹ In other words, the systems used some form of intelligent processing to automatically determine the fault(s) present in the monitored process and directly presented the machine's diagnosis to the human operator.

There is a kind of ambivalence in the systems surveyed, however. All of the systems attempted to provide the human operator with various sources of information about the monitored process or the intelligent system's processing to help the operator develop their own assessment. However, the systems generally did not include mechanisms explicitly designed to assist the operator deal with the data overload problems that can occur during fault management. Some of the human interface features in some of these systems may even increase the human operator's data overload problem.

The Selective Monitoring system (SELMON) under development directly addresses the information handling requirements of fault management and the danger of data overload (Doyle, Sellers and Atkinson, 1989; Doyle et al., 1990; Fayyad et al., 1990). This work uses machine intelligence in the cognitive tool style of interaction described in appendix C (figure C-3). The goal is to intelligently handle data coming from the monitored process to help the human operator focus on the subset that is significant for the current context (cf., also Woods and Elias, 1988; Woods, 1991).

¹ The systems examined in the case study usually contained many functions, only some of which were targeted at fault management.

Appendix C

Styles of Collaborative Interaction

What are the different ways that intelligent agents can interact in solving fault management problems? Figures C-1 through C-3 illustrate three basic styles of interaction between the human operator and an intelligent fault management system.

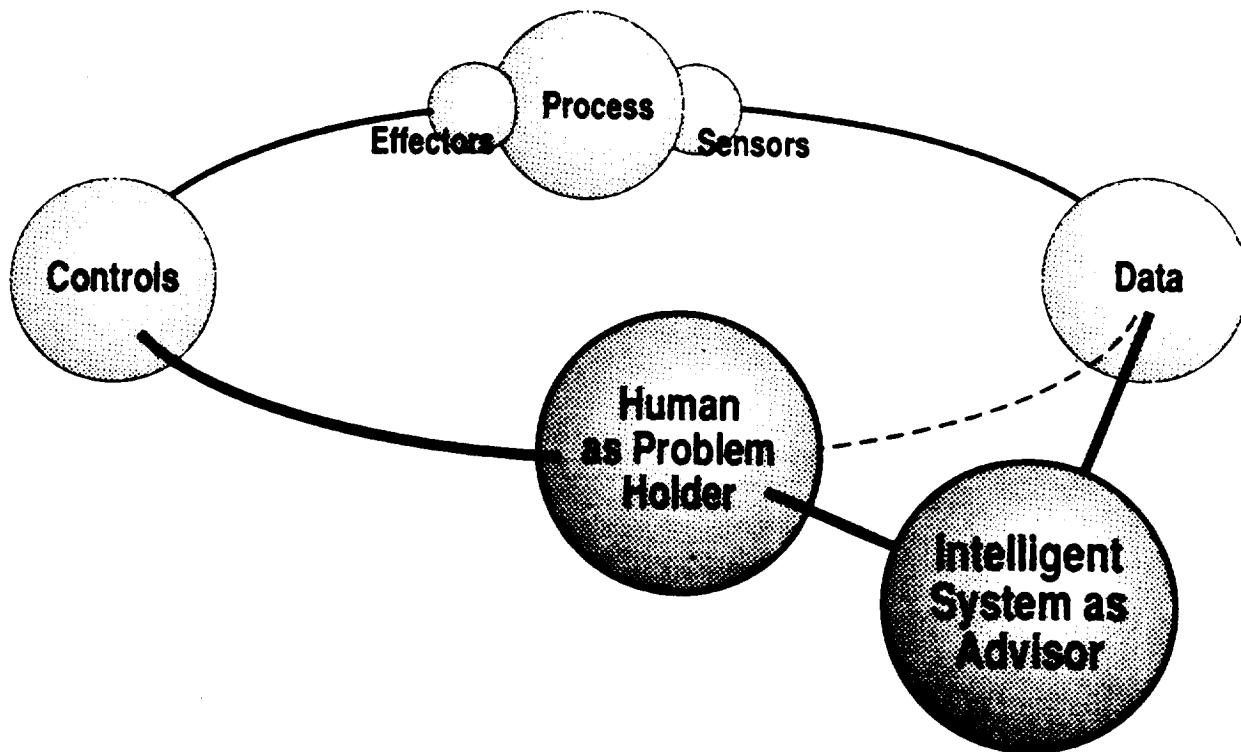
Figure C-1 illustrates the intelligent system as advisor architecture. As mentioned earlier, the first AI consultant systems needed the human to collect data. In aerospace fault management situations, of course, what happens is that the data on the state of the monitored process is available automatically to the intelligent system as well as to the human partner. While the intelligent system can monitor the engineered process directly, note that it cannot take any action independent of the human in the loop. In this paradigm, the human is the problem holder in that a person is always the one who has responsibility in these systems (cf., Woods, 1986 for a discussion of authority/responsibility relationships in human-intelligent computer interaction).

C.1 Advisory functions

What makes for a good advisor? There have been a variety of studies of human-human advisory interactions which can provide insight into the answer to that question (e.g., Coombs and Alty, 1981; cf., Woods and Roth, 1988 for a review). Advisory interactions where the advisor exclusively controls the interaction and then offers a solution for the problem holder are generally seen as unsatisfactory either in terms of performance (brittleness) or on some other dimensions (learning). Whereas in successful advisory interactions, there tends to be shared interaction, aid in identifying important facts, and greater focus on problem definition. This allows the problem holder to gain some expertise through the advisory process. The results suggest that good advice is more than recommended solutions. The following are different types of advisory interactions.

- Substitution or advisor as cognitive prosthesis

The kind of advisory interaction that we normally think of in the context of the typical AI consultant system we call the substitution or prosthesis approach. In this approach the human operator is seen as incompetent and the machine advisor substitutes as problem solver in order to improve performance (Woods et al., 1990). Note that the problem holder and advisor function as essentially independent problem solvers. However, in aerospace fault management, as in many other domains, the people in the system are highly trained, knowledgeable and often possess high levels of expertise themselves. Note how the substitution style represents a shift in locus of control from the problem holder to the advisor; see Woods (1986), Muir (1987), Billings (1990) for discussions of locus of control, responsibility/authority relationships and trust in human-intelligent system interaction.



© 1991 Woods, Potter, Johannesen, Holloway

Figure C-1. Styles of Human-Intelligent System Interaction: Intelligent System as Advisor and Human as Problem Holder

- **Reminding or broadening**

In this style the advisor acts to remind the human problem holder of relevant possibilities. The goal is not to output a possible answer, but rather to aid the human in developing their solution by ensuring a broad consideration of all possibly relevant factors or hypotheses or consequences. Performance is aided by (a) including more of the possibly relevant data in the problem solving process, (b) generating a larger, more thorough, set of candidate hypotheses that might account for the situation, (c) broadening consideration of the consequences that might follow from a chosen course of action. This approach targets several psychological classes of human errors, for example, errors of missing side effects.

- **Critiquing**

These systems analyze a user's decision, solution or plan of action in order to detect errors, verify the adequacy of the human's decision, or suggest improvements. Prototype critiquing systems, or machine critics, have been built for several domains, predominately medical applications (Langlotz and Shortliffe, 1983; Miller, 1986; Fischer et al., 1990). Note that the machine critic needs to be able to determine and analyze the human's decision or plan of action. Thus, it is related to work on user intent inferencing, user modeling and error detection (Rouse et al., 1987-88; Hollnagel, 1991) in human-computer interaction and intelligent tutoring.

- **Teaching**

If the advisor is more expert than the problem holder on a particular kind of problem, then an important advisory function may be teaching. In this case, a good outcome to the advisory interaction in addition to achieving a better solution to the local problem, would be for the problem holder to know more about how to handle that kind of situation better in the future (Gadd and Pople, 1990).

- **Informational**

In this case the advisor may function like a staff member who assists in information handling. For example, the problem holder is confronted with a multiple failure situation where the standard plans for handling each individual fault interact negatively. The advisor as staff assistant may help manage information retrieval collecting the background documents and analyses that underlie each standard plan to support the adaptive planning process.

- **Merging partial overlapping expertise**

What may be one the most important characteristics in aerospace advisory interactions is that frequently the advisor(s) and problem holder possess partial and overlapping expertise which must be integrated in order to solve the problem at hand (Jackson and Lefrere, 1984).

C.1.1 Formulating and Delivering Advice

Developing a style of interaction between human and intelligent advisor involves design decisions about how to formulate and deliver advice. One of these dimensions of advisory interactions has to do with the unit or grain of expression of the advice that should be offered at a given stage of an evolving incident. Should the system generate highly specific micro-assessments or micro-responses, or should the advice be formulated in terms of global control strategies or plans of action leaving the operator some degrees of freedom? Does the advisor

say "turn the knob two turns to the right" to compensate for low pressure in the system or does the advisor say "turn the heater system on"? A second dimension is when should the advisor interject during an evolving incident, for example, when does it interrupt the problem solver to remind him or her of another hypothesis or to critique the problem solver's actions. Third, should the intelligent system attempt to generate advice under all conditions or only for those situations where the appropriateness of the advice can be assured?

These design problems can be particularly difficult for tasks that involve continuously changing dynamic processes. For example, micro-advice about what response to make is antithetical to the formulation of flexible, adaptive responses that characterizes skilled performance, because no context or rationale is provided. This form of advice can inhibit the ability of the practitioner to formulate an adaptive response to an unanticipated situation. Thus, action oriented advice should be integrated with information about what process state the advice is a response to (the intelligent system's situation assessment) and with information about what are the expected effects of this response on the relevant parameters or system states within the monitored process.

To deal with design problems like the above Woods and Roth (1988; Roth and Woods, 1989) proposed two concepts for human-advisory system interaction. One is the concept of severity dependent variations in the grain of expression of advice. In this approach the intelligent advisor is designed to be quiescent when the system is relatively stable or normal and to gradually output stronger and more specific response advice as the situation worsens. In low exposure regions (exposure refers to the risk of negative consequences), advice takes a relatively unobtrusive form as broad dynamic targets on major parameters that allows the operator wide latitude and preserves his or her responsibility to formulate responses, for example, analog indications of the intelligent system's computation of desired state that can vary in width as well as in position. As the situation worsens, the advice becomes more pointed and increasingly restricts the operator's latitude.

This concept of a severity dependent grain of advice is an approach to interfacing machine advisor and human practitioner that balances several constraints on what is good advice: (a) deliver advice in those situations where one can generate clearly appropriate information, (b) but interject advice only when it is needed (which implies an understanding of the current state and likely future course of the problem solving process and which tends to be an inverse function of how easy it is to generate advice), and (c) preserve the operator's initiative and flexibility in responding to situations beyond the capacity of the advisor (cf., Roth et al., 1987).

Another issue in building advisory systems is the problem of explanation -- how to communicate to the user why the system has determined that the recommended action is needed and how the action it suggests will achieve the desired outcome. While the importance of explanation is generally recognized, advisory systems often fall short in this respect. Typically, explanation consists of a justification of the machine's decision in the form of a trace of the detailed process by which the machine generated the advice. One can call this retrospective explanation -- an explanation for some event (the intelligent system's diagnosis or suggested corrective action) that has already occurred (Wick and Thompson, 1989).

But notice how the human-in-the-loop must decide when to question the machine's advice and to call for an explanation. To do this the human problem holder must build and maintain his own assessment of the world. Frequently in intelligent advisory interactions, the human problem holder does this independently using whatever interface capabilities are available for examining the monitored process without help from or building on the cognitive work already performed by the intelligent system (e.g., Roth et al., 1987). Independent problem solving

and cross checking is a very weak style of cooperative problem solving for most applications, especially monitoring and controlling dynamic processes.

Another problem with relying exclusively on retrospective explanation in dynamic fault management applications is time pressure. The human problem solver must break away from examining the monitored process to evaluate the soundness of the intelligent system's advice at exactly the time when there is a failure in the system perhaps requiring safing actions or various operator cognitive activities (see appendix B on the disturbance management cognitive activity in dynamic fault management). While the person just begins to evaluate the appropriateness of the intelligent system's assessment of the situation and its recommendation, the disturbance chain started by the fault continues to propagate towards various levels of negative consequences. Experience in the nuclear industry with non-AI diagnostic systems has shown that such a style of interaction is very likely to fail as a form of cooperative problem solving -- operational personnel devote their attentional and cognitive resources directly to the monitored process and filter out "interruptions" from the "advisor." Note that the demand for interacting or communicating with the other team member in cooperative problem solving goes up at the same time that the difficulty or degree of threat of the problem goes up. This correlation creates a difficulty whenever the communication bandwidth is low or the cognitive workload of interaction and interpretation with the agent is high. This interaction is the source of the phenomenon termed clumsy automation by Earl Wiener (cf., Wiener, 1989 or Cook, Woods and Howie, 1990 for data on clumsy automation phenomenon in another application).

An alternative approach to explanation is to present advice in the context of the assessment of current (and future) state of the monitored process in order to create a common frame of reference for the person and intelligent system (Woods and Roth, 1988). By having access to the intelligent system's diagnostic search, intermediate conclusions, model-based expectations and situation assessment, the human problem solver is in a better position to evaluate the soundness of the advisor's conclusions and recommendations when they are reached. In contrast to retrospective explanations, "explanation" is integral to the presentation of advice, rather than a side function that must be explicitly requested.

This, combined with explicit indications of the expected future evolution of the monitored process (e.g., the expected effects of a recommended response on the relevant parameters or system states within the monitored process), allows the human practitioner to track the intelligent system's understanding of the monitored process' evolution and supports the operator need to understand and track the the evolution of the monitored process. The common frame of reference concept for explaining the behavior of intelligent machines is designed to avoid the ubiquitous problems associated with opaque advice (Roth et al., 1987; Suchman, 1987).

An important point to draw from this discussion is that an automated problem solving module that outputs best diagnosis or recommended actions is only one component of an effective advisory system. The design of advisory systems requires careful orchestration of state representation and advisory elements.

C.2 Intelligent Subordinate

Figure C-2 illustrates another major style of interaction between human and intelligent system where the latter functions as an intelligent subordinate and the former as supervisory controller. Note that the intelligent system, as compared to the advisory paradigm, has the capability to act autonomously on the monitored process (to be strict, the subordinate is semi-autonomous). Figure C-2 is drawn with multiple subordinate and monitored sub-processes to illustrate the situation in which several subordinates have the capability for autonomous action and where

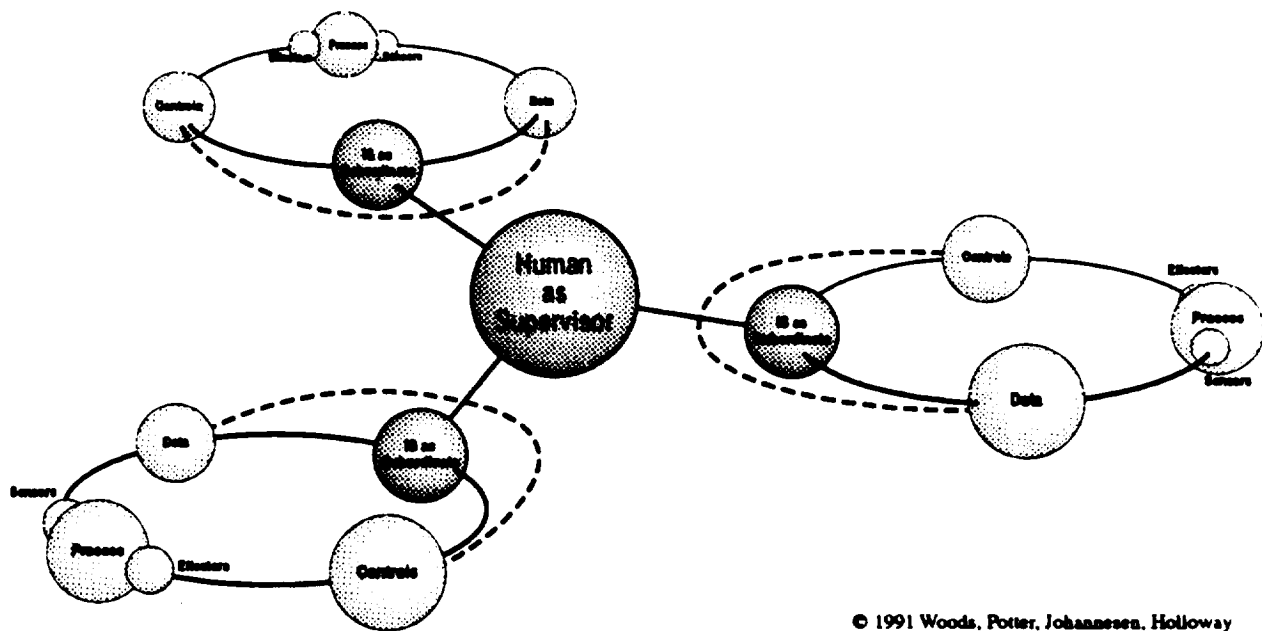


Figure C-2. Styles of Human-Intelligent System Interaction: Intelligent System as Subordinate and Human as Supervisory Controller

the human supervisor (second order control element) has responsibility over a larger part of the total monitored process (say, for example, all the power related related systems on the Space Station rather than just the power distribution system).

This style of interaction is very important in NASA space applications because of the need to increase system autonomy to reduce the human load in monitoring and managing Space Station systems as well as other aspects of space missions. But it is important to recognize that the human will still have the role of supervisor of a set of partially autonomous subordinates each with local scopes of responsibility.

C.2.1 Scope of Responsibility

Multiple cooperating agents can vary in scopes of responsibility (Roth and Woods, 1988). A scope of responsibility involves two components: a portion or area of the process where the agent is responsible for taking actions or supervising control, and a temporal horizon.

The scopes of responsibility of two or more agents can be independent but overlapping so that there is a coordination requirement. Each agent needs taking into account how events and changes within his own sphere can affect the processes within the other agent's scope of responsibility. Similarly, each agent needs to understand and anticipate how events and changes in the other sphere may affect his own scope of responsibility. Note we assume here that the cooperating agents have equivalent status.

A supervisor has a larger, encompassing scope of responsibility relative to a set of subordinates who work within narrower scopes. This entails periodic assessment of subordinate activities in relation to goal achievement and delegation to the subordinates; these tasks imply a shifting of the supervisor's attentional focus as events warrant.

Coordination from supervisor to subordinate focuses on delegation. Delegation involves assignment and removal/termination of delegated tasks either through intervention in subordinate's activities or manual takeover. Delegation implies that the delegator is actively in control and knowledgeable at some level about how to carry out tasks or how to achieve goals. The delegation role implies that the supervisor makes judgments about the capabilities of subordinates (where "capability" is a long term characteristic of the subordinate (cf., Muir, 1987)) and uses that assessment to decide when and how to delegate, and when to take over or intervene.

Supervisors exert strategic control. Supervisors coordinate subordinate activities at the boundaries of subordinate scopes of responsibility or in overlap areas; they need to re-direct subordinates when circumstances change (i.e. when new events occur or when new stages in the evolution of a scenario are reached).

Coordination from subordinate to supervisor focuses on when the subordinate should alert the supervisor to changes/events/trouble within the subordinate's scope of responsibility (see Norman, 1990; Sheridan, 1988; Billings, 1990 on coordination from subordinate to supervisor).

C.2.2 Control of Attention

One paradigmatic scenario for supervisor-subordinate interaction is a situation where the supervisor with his physical and mental resources devoted to some other area of the monitored process receives or recognizes some indication that there is some trouble in another part of the monitored process. The issue here is how to give the subordinate the ability to communicate, "Hey, there's something going on, maybe you want to know about this." This implies a

consideration of how the supervisor decides when to interrupt his ongoing line of work to investigate the activity or trouble report (Miyata and Norman, 1986). Note that in the China Air incident in commercial aviation the machine subordinate (the autopilot) did not tell the human supervisors or provide any clear indication that it was working harder and harder to keep the aircraft in the proper attitude until the disturbance (from an engine loss) grew too large for the capabilities of the subordinate, causing the aircraft to stall (Norman, 1990; Billings, 1990).

The broader scope of supervisory responsibility means that the supervisor cannot examine all parts of the monitored process within his or her scope simultaneously, at least not in detail. This raises the problem of alerting/shifting the supervisor's focus of attention; placing the supervisor in what Sorkin and Woods (1985), call the "alerted monitor" role. This attention switching process occurs in shifts from normal operations to upset conditions, from one part of the monitored process to another during abnormal operations, and from one kind of cognitive activity to another (e.g., interrupting diagnostic search in order to monitor for and verify expected subordinate responses or to take safing actions).

As part of the alerted monitor role, the supervisor needs to be able to quickly size up what is going on in the relevant portion of the process and integrate it with the larger context in order to dynamically prioritize and allocate his or her limited attentional resources (Gopher, in press). What resources are available to support the supervisor in answering: what kind of trouble is present? Where is the process headed (and how fast is the situation deteriorating)? What does the intelligent subordinate think is wrong in the system? What is the intelligent subordinate (or other autonomous agents) doing to stabilize (safing actions) or treat (corrective actions) the situation so far? What kind of trouble is present? Does the supervisor need to intervene, continue his other activities, or monitor the subordinate as it tries to handle the situation? Is the subordinate outside its range of capability?

Answering these situation assessment questions involves examining the monitored process and the intelligent system in an integrated fashion, i.e. knowing the state of the process and the state of control of the process. Do the interface and communication capabilities between the human supervisor and both the monitored process and the intelligent system support mentally economical, quick assessments or do they force the person to use slow, effortful deliberative processing. Another issue is whether the supervisor can trust the subordinate to be able to handle the situation. This relates to the supervisor's sense of confidence in the subordinate's competency (i.e. knowing the range of problems it can handle). This judgement is built up over a longer term pattern of interaction (Muir, 1987). There is data that people form judgements about the capabilities of automatic systems as they do of other people (Roth and Woods, 1988); they form judgements about the kinds of situations subordinates can handle on their own and the kinds of situations that will require take over or intervention because they lie outside their range of competency.

C.2.3 Clumsy Automation

Clumsy automation refers to the benefits of the automation accruing during workload troughs and the costs of automation (i.e., additional tasks, forcing the user to adopt new strategies, new communication burdens, new attentional demands) occurring during periods of peak workload, high criticality or high tempo operations. (e.g., Wiener, 1989; Cook et al., 1990).

The concept is based on the fact that in complex systems human activity ebbs and flows, with periods of lower activity, more self paced tasks interspersed with busy high tempo operations where task performance is more critical (Rochlin, et al., 1987). Machine automation is designed to shift workload from the human to the machine. But the critical design feature for well integrated cooperative work between the automation and the human is not some overall or

time averaged workload reduction, but rather how the automation impacts on low workload and high workload periods. Clumsy automation is an example of poorly coordinated cooperative work where the automation helps at times of least need and hinders performance in situations of greatest need.

There is a growing body of evidence that clumsy automation is a significant danger when new technology is introduced into many high consequence, complex domains. Questionnaire and interview results of new computer based commercial aircraft avionics ("glass cockpits") and a protocol analysis study of physician performance with new computer based operating room devices revealed cases of clumsy automation where the new device actually increased human workload during peak workload times and decreased it during less demanding periods (Wiener, 1989; Cook et al., 1990).

C.2.4 New monitoring requirements

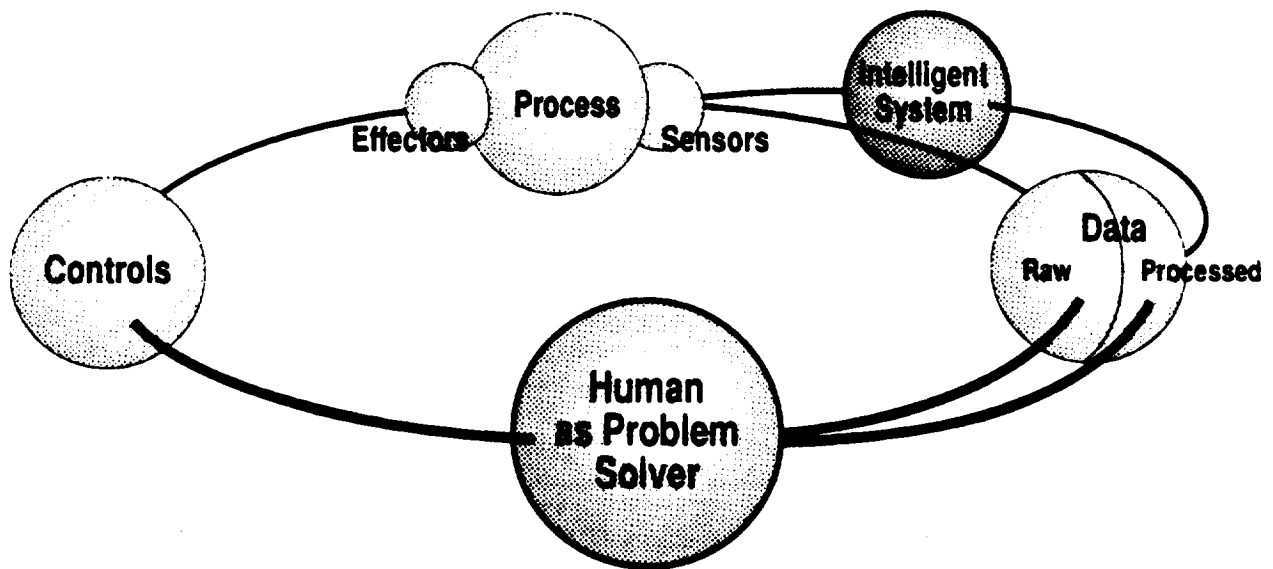
In the supervisory control style of interaction the human is removed from direct contact with the monitored process at least for nominal operations. While the supervisor's control actions may be reduced, this style of interaction tends to create new monitoring requirements. The supervisor still needs to know what the subordinate is doing (at some level), and what the state of control of the process is. Wiener (1989) found that, in reference to the automation, the most common questions asked by the supervisor (pilot) on highly automated commercial aircraft are: what is it doing? why is it doing that? what will it do next? The ability of the person to answer these three questions is the critical test for assessing the interface between the human and the intelligent system. Again, the theme of visualization is important -- in this case, visualization of intelligent system assessments/activities.

The supervisor should be able to actively search to assess subordinate activities in relation to the state of the monitored process rather than only wait for alerts from process or subordinate. This concept can be called the directed telescope (after van Creveld's, 1985, analysis of successful organizational problem solving in the military). Information about subordinate activities in relation to the state of the world should be continuously available rather than available only as part of retrospective explanation. For example, commercial aviation flightdecks integrate control information through throttle movements and yoke positions so that one agent (human pilot or autopilot) can know about the activities of other agents. This example also shows how the common frame of reference concept for advisor-problem holder interactions can be extended to supervisory-subordinate interactions.

C.3 Cognitive Tools

There is a third style for the interaction of human and intelligent system -- intelligent system as a cognitive tool wielded by the human problem solver (cf., Woods, 1986; Roth et al., 1987; Woods and Roth, 1988; Norman, 1990 for treatments of the cognitive tool approach). The cognitive tool approach is often overlooked because the AI research agenda tends to focus our attention on intelligent capabilities as an autonomous machine agent, where human-intelligent system interaction is like the interaction between two people.

Figure C-3 shows that in the cognitive tool approach the intelligent system is considered as just another source of data for the human problem solver. The intelligent system's power is applied to produce better (e.g., processed) information for the human operator (smart displays), support information management, and help the operator overcome data overload. Instead of thinking of the intelligent system's output as an "answer" to the problem, the cognitive tool metaphor suggests that we should think of the intelligent system's output as information for the



© 1991 Woods, Potter, Johannessen, Holloway

Figure C-3. Styles of Human-Intelligent System Interaction: Intelligent System as Cognitive Tool for Human Problem Solver

human problem solver, albeit much more processed and potentially more "pointed" (focused on possible actions) than measured data from the process, but still as information that needs to be gathered and integrated with other sources of information about the monitored process to produce an overall situation assessment.

This point of view helps explain some of the experience to date with developing automated fault diagnostic systems. Attempts to develop automated diagnostic systems focus on the intelligent system getting the right answer. However, the complexities of fault management make this very difficult for all possible circumstances that may arise. But from the point of view of the human operator, the diagnostic system is but one of many information channels that he or she must monitor, process and integrate to assess process state and meet process goals. Very often, the human operator's problem in fault management is coping with data overload so that attempts to add another information channel -- the diagnostic system -- that is not integrated with the other available sources of data have foundered by exacerbating the operator's data overload problem (Woods, 1989).

The cognitive tool approach emphasizes the active role of the person in the system and the importance helping him or her visualize what is going on in the monitored process and in the control of the process. This is, of course, is ultimate goal -- improving the management of faults, the diagnosis of faults and recovering from faults.

In several of the NASA intelligent system development cases examined, the resulting system functioned as a cognitive tool in the hands of an expert practitioner (Muratore et al., 1990; Intelligent Launch Decision Support System or ILDSS; Spacecraft Health Automated Reasoning Prototype or SHARP and potentially the Selective Monitoring system or SELMON). It was very difficult in these cases to untangle the benefits that accrued from better visualization of the monitored process, from better information handling capabilities that avoided data overload problems, and from conceptualization aids that allowed the user search for and discover patterns in the data. This reflects a general trend to focus more on supporting human interaction with the monitored process, through the medium of the computer, rather than just to develop stand alone machine consultants. As a result, the necessary human support functions become the critical driving force in design, where intelligent data processing becomes a means for implementing the desired functions (e.g., Woods and Elias, 1988; Doyle et al., 1990).

C.4 Summary: Styles of Human Interaction with Intelligent Systems

All three of the styles of interaction described above point to a common theme. In order to develop meaningful cooperation between human and intelligent machine, one cannot simply build a stand alone intelligent system and then decorate it with human-computer interface features. Integrating human and machine problem solvers into an effective cooperative system requires serious consideration of the desired coupling between human and intelligent system as an integral part of the design of intelligent system itself. The concept developed for how the intelligent system will assist the person can strongly constrain the architecture and design of the intelligent system itself. When one first develops an autonomous machine problem solver and, only later, considers how the person will use the system to achieve better performance, the resulting design often fails to make allowances for features that turn out to be critical for people to make effective use of the system's capabilities.

The three styles of human interaction with intelligent systems described above are intended as heuristics to guide our thinking about how to couple human and machine intelligence. A given intelligent system is very likely to have some characteristics from each of these styles. Each style is one point of view on effective collaboration and real world success stories probably will need to use some aspects of each style.

Appendix D

Description of Fault Management Agent Activities and Information

D.1 Fault Management Agent Activities

In section 3.3.1, three types of agent activities were identified:

- Monitoring and Assessment
- Planning and Dynamic Re-Plan
- Intervention and Control

Additionally, Coordination activities were identified for multi-agent systems. These categories of activities have been used to organize the agent activities required for fault management.

Monitoring and Assessment

- Assess state, status, health, and configuration to determine behavior of monitored process, related peripheral systems, and the intelligent system
- Compare on-going operations to planned activities
This activity requires access to flight procedures and the schedule of crew activities.
- Monitor the activities of the fault management team (intelligent system, crew, ground controllers) for errors and anomalous behavior
- Assess the accuracy and applicability of intelligent system conclusions
The human must understand intelligent system conclusions and integrate them with other information being monitored to make final decisions in fault management situations.
- Distinguish between nominal behavior and anomalous behavior
It is necessary to discern anomalous behavior to identify when faults occur and to verify that faults have been corrected. Parameters required for assessment of behavior include state, status, health, configuration, on-going operations, and indicators of anomalies (e.g., alarms)
- Relate anomalous behavior and failures to loss of functional capability and hardware items
This function requires access to an analysis of fault modes and effects of faults, design specifications, and functional descriptions.
- Relate loss of functional capability to resulting impact to safety or mission goals
Safety and mission impacts due to functional loss are dependent upon the criticality of the lost capability, the availability of redundant capability, and the ability to satisfy mandatory flight conditions specified in the flight rules and mission objectives.
- Assess Go/No Go calls based on impacts and remaining capability
Go/No Go calls are status checks of primary monitored systems conducted prior to a critical activity that has global impact to assess readiness to perform that activity. These calls are based on an assessment of the current capability to perform the activity and must consider any impacts introduced by that activity due to functional losses. The criteria for these calls are outlined in the flight rules.

- **Identify possible faults resulting in observed anomalous behavior**
To initiate fault diagnosis, candidate faults must be identified for investigation. An analysis of failure modes and effects, design specifications, and functional descriptions can be useful in postulating faults. Groups of indistinguishable faults (i.e., fault ambiguity groups) within these suspected faults should also be identified.
- **Assess the reliability and quality of information from both the monitored process and the intelligent system**
The reliability of information is based on an assessment of confidence in source of the information. The quality of information is related to imperfections in the information.
- **Assess the availability of data**
Sampled data that are not changing due to loss of the data source (i.e., unavailable data) should be distinguishable from data that are not changing due to steady-state or stable conditions (i.e., unchanging but available data).

Planning and Dynamic Re-Plan

- **Assess potential for a failure to propagate and affect other portions of the monitored process or associated peripheral systems**
Failure propagation potential is considered when evaluating reconfiguration options. Propagation effects are determined for all suspected faults. This determination requires an understanding of the causal connectivity within the affected portion of the monitored process or associated peripheral systems.
- **Determine reconfiguration options or schedule changes to minimize impact of anomalous behavior**
Options for reconfiguration or rescheduling depend upon the ability to continue on-going operations and conduct planned operations with the remaining functional capability while minimizing the impacts to mission and meeting safety requirements. The potential for degradation of capability due to failure propagation and the criticality of failures must also be considered.
- **Determine options to reconfigure the intelligent system when in error**
Reconfiguration of the intelligent system can include such options as changing the mode of operation, selective disabling of portions of the knowledge base, loading of alternate knowledge bases, or a complete reset to initial conditions.
- **Assess the impact of operator intervention into or take over of either the monitored process or intelligent system prior to taking action**
Safety and mission impacts due to operator intervention into the monitored process or the intelligent system should be assessed prior to the execution of intervention actions.
- **Predict the anomalous behavior associated with a fault**
A common method of identifying faults is to postulate a fault and see if the predicted behavior matches actual behavior of the affected system. An analysis of failure modes and effects, design specifications, and functional descriptions are used to determine the behavioral characteristics of the affected systems resulting from a postulated fault.

- Explore alternatives for reconfiguration and schedule changes after functional loss
Information required to explore reconfiguration and schedule changes includes an assessment of the remaining capability and its criticality to continued operations and the impacts of changes on mission goals and objectives.
- Explore alternatives for recovery after identification of fault
The criticality of recovering lost capability must be balanced against the impacts and risks of the recovery process.
- Assess recovery options based on identified fault
Once a fault has been identified, the ability to recover from the fault must be assessed. Malfunction correction procedures execute recovery options and are selected on the basis of achieving the required functionality to accomplish mission objectives safely.

Intervention and Control

- Control the Intelligent system
Control of the intelligent system refers to such functions as starting and stopping, restarting from stored state, and playback using recorded data.
- Redirect the intelligent system
Redirection of the intelligent system reasoning process can include:
 - Correcting intermediate intelligent system solutions
 - Postulating likely alternatives and selecting between hypotheses
 - Altering information internal to intelligent system processing
 - Controlling activation of portions of knowledge base
- Control the acquisition of data
Acquisition of data requires execution of data collection software, data recording, data quality assessment, and entering information unavailable from data collection software (i.e., operator input).
- Command monitored process
Command options include initiating action in the monitored process or altering the knowledge or data in monitored process. These options can either be exercised by the crew onboard the vehicle or by command uplink from the ground.
- Correct spurious or erroneous information from monitored process
Capabilities to correct information include filtering data, discarding data outside limits, disabling use of data, or selecting an alternate source of the information. Simulation can be used to estimate data is not available from any source.
- Initiate scheduled procedures
- Alter scheduled activities in response to fault management activities
- Establish alternate modes of operation for both intelligent system and monitored process
The selection of modes of operation will be dependent upon the current activity and the roles defined for each member of the fault management team.

- Select between alternative solutions for:
 - Reconfiguration and schedule changes after functional loss
 - Recovery after identification of fault
- Takeover of intelligent system responsibility by operator

Two types of takeover are possible. Partial takeover allows the operator to selectively influence performance of the activity (e.g., enable/disable rule base). Override is a complete takeover of the activity.
- Take over of monitored process (crew command, uplink)

Take over of the monitored process is accomplished in the same way as intervention, via crew commanding or command uplink from the ground.

Coordination

- Clarify intelligent system actions and conclusions by inspection of relevant information at intermediate steps of the reasoning process

Relevant information includes internal states, intermediate solutions, alternatives and hypotheses, and contents of the knowledge base.
- Review of algorithms, processes, and reasoning strategies used to compute or derive information about the intelligent system and the monitored process

These strategies are defined in the design specifications and functional requirements for the system of interest. Forms that this information could take include schematics, knowledge representation, and functional diagrams.
- Evaluate data trends and system performance for both intelligent system and monitored process

This evaluation requires access to data archives, performance specifications, and mission and design constraints for both the intelligent system and the monitored process. The evaluation should take into account any constraints imposed by the mission.
- Clarify agent responsibilities and task assignments
- Review previous actions of fault management team in the context of current events

This requires access to archived events and actions of the fault management team.
- Synchronization of dependent activities (e.g., wait for a result)
- Archival of important information and provision for access to that information at a later time

Potential uses include re-execution of intelligent system, review of important information from either intelligent system or monitored process, and identification of trends.
- Manipulation of archived information from both the monitored process and intelligent system

Manipulation can be as simple as review of information or as complex as processing data using a simulation or trend analysis program. Reviewing the results of this manipulation can be accomplished either by viewing the displays used for real-time support or by providing display formats specific to the manipulation.

- Alteration of the appearance of the display to meet information needs
The ability to access information available but not currently visible (e.g., review archived data using special display formats) requires some means of navigating through the available display formats.

D.2 Fault Management Information

In section 3.2.3, four types of information were identified based on the source of information:

- Dynamic Information
- Baseline Operations Information
- Mission Specific Operations Information
- Design knowledge

These types of information have been used to organize the information required for fault management.

Dynamic Information

- **Sensed and Processed Data**
This is a very large class of information, since it includes much of the telemetry downlisted from the vehicle and the ground-based trajectory data. This information includes both raw telemetry (e.g., sensor measurements) and data computed on the ground (e.g., Space Shuttle COMPS processing).
- **Time**
Time has been distinguished as a separate category of sensed information to emphasize its function as the reference point for synchronizing information. Each piece of dynamic information will have an associated timetag. Examples of the variety of types of timing information include current time, elapsed time, time of an event, and the ordering of timed events.
- **System State, Status, and Configuration**
State, status, and configuration are required for the primary monitored system, peripheral systems, including the communications path, and the intelligent system.

State information evaluates the current condition of the system being assessed (e.g., Remote Manipulator System (RMS) is DEPLOYED). Information relevant to intelligent system state includes the intermediate conclusions and alternative hypotheses and solutions encountered in transitioning between states.

The Status of a system is an evaluation of the health of the system (e.g., a motor on the RMS Manipulator Positioning Mechanism has failed). Intelligent system status may not be explicitly stated. A related research issue is the ability for an intelligent system to describe its limitations and point out when it encounters problems outside its expertise (Wexelblat, 1989).

The Configuration of a system is an assessment of the system's readiness to perform a specified task based on the state and status of the system (e.g, RMS Powerup is complete and RMS is ready for Checkout). The Configuration of the intelligent system would include such information as the knowledge bases that are enabled or the model currently employed. Reconfiguration is the process of changing a system's configuration in response to a new situation. Misconfiguration is the process of incorrectly altering a system's configuration for a specified task.

- **Alarms, Anomalies, Faults, and Failures, including suspected faults and fault ambiguity groups**

Anomalies are irregular system behavior or conditions that can be caused by a variety of influences, including faults, environmental factors, operator error, etc. A fault is the cause of failure in a system, subsystem, component, or part. A failure is the inability of a system, subsystem, component, or part to perform its required function within the specified conditions, (i.e., lost capability) (JSC, January 1989). For example, one anomaly of a Display and Control panel is that a light did not glow when expected. The resulting failure is the inability of that panel light to glow and the fault is a burned out light bulb. Alarms are often ranked according to the severity of the failure that they indicate. For example, alarms indicating emergencies can be distinguished from alarms indicating that parameter value is outside expected limits. An example of the alarm ranking used for Space Shuttle malfunctions is (JSC, June 1990):

 - class 1: Emergency
 - class 2: Caution and Warning (C&W)
 - class 3: Alert from the Major Functions (GNC or Systems Management)
 - class 0: Limit Sense

Alarms do not always translate directly into a fault. Ambiguous faults are faults that cannot be distinguished using the available symptoms. Misconfigurations and errors on sensed data can result in alarms that may be misidentified as faults with the same symptoms. Faults can occur sporadically or inconsistently (i.e., intermittent faults) which complicates the fault identification process.
- **Mission Impacts and Failure Propagation Potential**

Failures can result in impacts to crew, vehicle, and mission. These impacts have implications for the safety of the crew and vehicle and can affect both on-going and scheduled operations. Failure impacts include not only immediate effects, but the potential for a failure to propagate (i.e., cause other forms of aberrant behavior) over time.
- **Context of On-going Operations**

Operational context is the setting or conditions that represent capabilities and constraints implicit in on-going operations. Examples of common operational contexts are (1) crew or vehicle activity (e.g., maneuver in progress), (2) configuration (e.g., RMS ready for checkout), (3) capability (e.g., telemetry unavailable after Loss Of Signal).
- **Go/No Go Calls**

Go/No Go calls are status checks of all primary monitored systems conducted prior to a critical activity to assess readiness to perform that activity.

- **Functional Capability**
Functional capability is the ability to perform the pre-defined functions of a system. An assessment of functional capability includes both the criticality of the function and the availability of redundant capability. After a functional loss, the remaining functional capability is assessed.
- **Operator Inputs**
Operator inputs include commands to the monitored process (e.g., either by crew keypad entry or ground uplink), control inputs to the data acquisition system, and inputs to the intelligent system. The inputs to the intelligent system can include information unavailable in electronic form (e.g., voice, not on downlist), control and redirection of the intelligent system, and configuration of the human-computer interface.
- **Performance of the monitored process, related peripheral systems (e.g., sensors) and the intelligent system**
- **Trends in archived data**
Archives are analyzed to identify trends in the occurrence of faults and failures of the monitored processes, sensors, and the intelligent system. Performance trends are also analyzed.
- **Dynamic intelligent system information, including:**
 - State, status, health, and configuration
 - Internal states
 - Hypotheses and intermediate or alternative solutions
 - Activities (previous, current, planned)
 - Reliability assessment
- **Archived Information**
Archived information is information recorded for later use. Types of data archives frequently made include:
 - Telemetry and Trajectory (e.g., Space Shuttle Data Tabs; JSC, October 1983)
 - Processed information (e.g., output of intelligent system)
 - Internal state information (e.g., intermediate states and alternatives)
 - Checkpoint
 - Actions taken by fault management team
 - Annotations from fault management team (e.g., operator activity logs)
 - Hardcopy of displayed information

Mission Specific Information

- **Mission Goals**
The objectives of a planned mission. For Space Shuttle, these objectives are outlined in the Flight Data File (FDF).
- **Schedules and Activity Timelines**
Time-ordered sequences of activities that were previously scheduled to meet mission goals. Scheduled activities can be altered during the mission in response to anomalies. An example from Space Shuttle is the Crew Activity Plan (CAP).

- **Mission Constraints**
Mission constraints define the range of valid values and the boundary conditions levied by specific mission requirements (e.g., a unique limit used for limit sensing, special performance requirements).
- **Mission Specific Procedures**
Mission Specific Procedures are special procedures required to achieve mission goals and accommodate mission constraints.
- **Mission Configuration**
Mission configuration defines the readiness of the monitored process and peripheral systems to perform mission-specific tasks. Mission configuration includes such information as values for initialization that are loaded prior to a mission.
- **Mission Specific Expertise and Experience**
Information acquired during training in mission specific simulations and previous support of similar missions

Baseline Operations Information

- **Flight Rules**
Flight rules document nominal and off-nominal conditions and behavior and constraints of monitored processes (e.g., the Delta State Update limits specify conditions for performing an update of the onboard navigation state with the redundant ground state).
- **Procedures**
Procedures are methods for accomplishing mission goals within the constraints of flight rules. Procedures encountered during the case study include:
 - Nominal activities
 - Test and Checkout
 - Safing
 - Reconfiguration
 - Malfunction Diagnosis and Correction
 Procedures not only define actions but include the behavior expected to result from those actions.
- **Modes of Operation**
The monitored process can have modes of operation that define the operating constraints of the system (e.g., Space Shuttle OPS modes indicate the flight software load which is unique to the phase of support (e.g., ascent, on-orbit, de-orbit, entry)). For the intelligent system, modes of operation result from the defined roles and responsibilities the team members.
- **Operational Performance Requirements**
Performance specifications based on operational requirements.
- **Failure Signatures**
Failure signatures are distinctive patterns of information that uniquely identify a failure. These patterns are usually recognized as a result of operational experience.

- **Operations Expertise and Experience**
Information acquired during training in generic simulations (i.e., simulations independent of a specific mission objective) and during actual flight support. This expertise can include operator heuristics and unwritten procedures.

Design Knowledge

- **Monitored System Specifications**
The monitored system is specified by functional descriptions (e.g., requirements, flow charts, algorithms), physical descriptions (e.g., location, connectivity), and design specifications (e.g., schematics).
- **Intelligent System Design Specifications**
The intelligent system design specifies the reasoning strategies employed, defines the information contained in the knowledge base, and describes the knowledge representation.
- **Design Constraints**
Design constraints define the range of valid values and the boundary conditions resultant from the system design (e.g., design constants).
- **Design Performance Specifications**
Performance specifications based on the design.
- **Failure Modes and Effects**
Failure effects are the effects of failures on functional capability and failure modes are the resultant operating modes. Both types of information are based on the design of vehicle systems (i.e., the monitored processes). For Space Shuttle, this information for hardware systems is captured in the Failure Modes and Effects Analysis (FMEA) documentation.
- **Redundancy**
The availability of alternate, comparable capability. Redundancy is a typical approach to risk management in space systems.
- **Criticality Ratings**
Criticality rating classifies hardware items and functional capability by the potential effect of a failure on the mission and safety. Functional criticality indicates the effect of the loss of all redundant capability. Hardware criticality indicates the effect of a hardware failure with available redundancy. For Space Shuttle, Critical Item Lists (CIL) are constructed that identify the single point failures and redundant elements with respect to safety and mission (JSC, January 1989).
- **Design Expertise**
Expertise acquired during the design (e.g., design engineers) and implementation (e.g., implementing contractor) of the monitored process or the intelligent system.

References

- Abbott, K. (1988), "Robust Operative Diagnosis as Problem Solving in a Hypothesis Space", *Proceedings of the Seventh National Conference on Artificial Intelligence*, American Association for Artificial Intelligence.
- Abbott, K. (1991), in press.
- Arkes, H.R., and K.R. Hammond (1988), *Judgment and Decision Making: An Interdisciplinary Reader*, Cambridge University Press.
- Backes, P., T. Lee, and K. Tso (February, 1990), "Teleroobotics System Technology: Final Report, FY '89 (draft)", *Proceedings of the Space Station Evolution Symposium*, Volume 2, Part 2, NCP 10044, Johnson Space Center, Houston, TX: NASA.
- Balkanski, Cecile (1990), "Modelling Act-Type Relations in Collaborative Activity", TR-23-90, Center for Research in Computing Technology, Harvard University.
- Becker, R. A. and W. S. Cleveland (1984), *Brushing the Scatterplot Matrix: High-Interaction Graphical Methods for Analyzing Multidimensional Data*, Technical Report, AT&T Bell Laboratories Technical Memorandum.
- Billings, C. E. (1990), *Human-centered Aircraft Automation Philosophy: Rationale and Concepts Guidelines*, Technical Report RTOP 505-67-21, Ames Research Center: NASA.
- Bloom, Charles P. (April, 1991), discussion at Human Factors in Computing Systems Conference, CHI '91 Workshop: Advances in Computer-Human Interaction in Complex Systems, Minneapolis, MN: Honeywell, Inc.
- Bodker, Susanne (1989), "A Human Activity Approach to User Interfaces", *Human-Computer Interaction*, Volume 4, pp. 171-195.
- Britten, Julie and Vivian Hardwick (December, 1986), "System Malfunction Procedures Requirements", Appendix F of Space Shuttle Crew Procedures Management Plan, Mission Operations Directorate, Johnson Space Center, Houston, TX: NASA.
- Brooks, Ruven (April, 1991), discussion at Human Factors in Computing Systems Conference, CHI '91 Workshop: Advances in Computer-Human Interaction in Complex Systems, Austin, TX: Schlumberger Laboratory for Computer Science.
- Brown, Daryl and Tom Kalvelage (September, 1988), *INCO Expert System Project Real Time Data System User's Guide*, Mission Operations Directorate, Johnson Space Center, Houston, TX: NASA.
- Buck, Leslie (November, 1989), "Human Operators and Real-Time Expert Systems", *Expert Systems*, Volume 6, Number 4.
- Chandrasekaran, B., M.C. Tanner, and J.R. Josephson (spring, 1989), "Explaining Control Strategies in Problem Solving", *IEEE Expert*, pp. 9-24.
- Chu, Rose (June, 1990), demonstration of prototype at the Eighth Users' Conference for the Transportable Applications Environment (TAE), Atlanta, GA: Georgia Institute of Technology.

- Cook, R. I., D. D. Woods, and M. B. Howie (1990), "The Natural History of Introducing New Information Technology into a Dynamic High-risk Environment", *Proceedings of the Human Factors Society 34th Annual Meeting*.
- Cook, R. I., D. D. Woods, E. McColligan, M. B. Howie (June, 1990), "Cognitive Consequences of "Clumsy" Automation on High Workload, High Consequence Human Performance", *SOAR 90, Space Operations, Applications and Research Symposium*, Johnson Space Center, Houston, TX: NASA.
- Coombs, M. J. and J. L. Alty, (1980), "Face-to-face Guidance of University Computer Users --II: Characterizing Advisory Interactions", *International Journal of Man-Machine Studies*, 12:407--429.
- Czerwinski, M. P. (1990), review of PDRS HCI design concepts, Houston, TX: Lockheed Engineering and Sciences Company.
- Czerwinski, M. P. (1990), "Research Issues in Interfaces to Intelligent Systems", Houston, TX: Lockheed Engineering and Sciences.
- Czerwinski, M. P. (1991), "Automated System Function Allocation and Display Format", submitted for publication.
- de Kleer, J. and J. Brown (1984), "A Qualitative Physics based on Confluences", *Artificial Intelligence*, No. 24.
- Doyle, R. J., L. K. Charest, L. P. Falcone, and K. Kandt (July, 1990), "Addressing Information Overload in the Monitoring of Complex Physical Systems", *Fourth International Qualitative Physics Workshop*.
- Doyle, R., S. Sellers, and D. Atkinson (1989), "A Focused, Context Sensitive Approach to Monitoring", *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, IJCAI.
- Elm W. C. and D. D. Woods (1985), "Getting Lost: A Case Study in Interface Design", *Proceedings of the Human Factors Society 29th Annual Meeting*.
- Farry, Kristen (August, 1987), draft data flow charts for Space Station fault management, Houston, TX: Rockwell Shuttle Operations Co.
- Fayyad, U. M., D. Berleant, L. K. Charest, R. J. Doyle, L. P. Falcone, K. Kandt, and G. T. Tsuyuki (February, 1990), *Selective Processing in Monitoring Autumn 1989 Report*, Technical Report, JPL D-7103, Jet Propulsion Laboratory, .
- Fischer, Gerhard (January, 1989), "Human-Computer Interaction Software: Lessons Learned, Challenges Ahead", *IEEE Software*, pp. 44-52.
- Fischer, G., A. Lemke, and T. Mastaglio (1990), "Using Critics to Empower Users", *Proceedings of the CHI'90 Human Factors in Computing Systems Conference*, Association for Computing Machinery.
- Forbus, Kenneth D. (1984), "Qualitative Process Theory", *Artificial Intelligence*, No. 24.
- Forbus, Kenneth D. (February, 1991), *Qualitative Physics as a Tool for Human-Computer Interaction*, The Institute for the Learning Sciences, Evanston, IL: Northwestern University.

Gadd, C. S. and H. E. Pople, Jr. (1990), "Evidence from Internal Medicine Teaching Rounds of the Multiple Roles of Diagnosis in the Transmission and Testing of Medical Expertise", *Diagnostic Monitoring of Skill and Knowledge Acquisition*. N. Frederiksen, R. Glaser, A. Lesgold and M. G. Shafto, editors, Hillsdale, NJ: Erlbaum.

Gasser, Les (January, 1991), "Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems Semantics", *Artificial Intelligence*, volume 47, Numbers 1-3, pp. 107-138.

Gentner, D., and A. Stevens (1983), *Mental Models*, LEA Associates, Inc.

Gibson, J. J. (1979), *The Ecological Approach to Visual Perception*, Houghton Mifflin, Boston.

Gnabasik, Mark (February, 1990), discussion of other PDRS prototypes, Houston, TX: The MITRE Corporation.

Gopher, D. (1991), "The Skill of Attention Control: Acquisition and Execution of Attention Strategies", *Attention and Performance XIV*, Hillsdale NJ: Erlbaum, in press.

Hefley, Bill (April, 1991), discussion at Human Factors in Computing Systems Conference, CHI '91 Workshop: Advances in Computer-Human Interaction in Complex Systems, Pittsburgh, PA: Carnegie Mellon University.

Henderson A. and S. Card (1987), "Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-based Graphical Interface", *ACM Transactions on Graphics*, 5:211--243.

Herrman, C.T. (October, 1988), *Visual Language Systems for Computer Displays*, WP-88W424, McLean, VA: The MITRE Corporation.

Hewitt, Carl (January, 1991), "Open Information Systems Semantics for Distributed Artificial Intelligence", *Artificial Intelligence*, volume 47, Numbers 1-3, pp. 79-106.

Hollnagel, Erik (June, 1990), "The Design of Error Tolerant Interfaces", *Proceedings of ESA Symposium: Ground Data Systems for Spacecraft Control*, pp. 511-514.

Hollnagel, E. (1991), "The Genotype and Phenotype of Erroneous Action: Implications for HCI Design", J. Alty and G. Weir, editors, *Human-Computer Interaction and Complex Systems*, London: Academic Press, in press.

Jackson, P. and P. Lefrere (1984), "On the Application of Rule-based Techniques to the Design of Advice Giving Systems", *International Journal of Man-Machine Studies*, 20:63--86.

Jakob, Francois and Pierre Suslenschi (April, 1990), "Situation Assessment for Process Control", *IEEE Expert*, pp. 49-59.

Johns, Gordon L., (September, 1990), *Graphic Interfaces to Intelligent Fault Management Systems: Issues and Guidelines*, MTR-90W00103, Houston, TX: The MITRE Corporation.

Jones, William P. (April, 1991), discussion at Human Factors in Computing Systems Conference, CHI '91 Workshop: Advances in Computer-Human Interaction in Complex Systems, Seattle, WA: Boeing Computer Services.

JSC (October, 1983), *PDRS Console Handbook, Volume I*, JSC-17525, Johnson Space Center, Houston, TX: NASA.

JSC (April, 1987), "MOD Drafting Standard", Mission Operations Directorate.

JSC (1988), *RMS Mission Histories*, Mission Operations Directorate, Johnson Space Center, Houston, TX: NASA.

JSC (February, 1988), *Payload Deployment and Retrieval System Overview Workbook*, PDRS OV 2102, TD383, Johnson Space Center, Houston, TX: NASA.

JSC (April 1988), *PDRS Malfunction Workbook*, PDRS MAL 2102, TD385, Johnson Space Center, Houston, TX: NASA.

JSC (May, 1988), *Space Station Information System Human-Computer Interface Guide*, version 2.0, USE-1000, Johnson Space Center, Houston, TX: NASA.

JSC (December, 1988), *Payload Deployment and Retrieval System Nominal Operations*, PDRS NOM OPS 2102, TD386, Johnson Space Center, Houston, TX: NASA.

JSC (January, 1989), *Remote Manipulator System Failure Modes and Effects Analysis and Critical Items List*, JSC-22373, Johnson Space Center, Houston, TX: NASA.

JSC (February, 1989), *PDRS Operations Checklist, STS-31 Flight Supplement*, JSC-16987-31, Johnson Space Center, Houston, TX: NASA.

JSC (December, 1989), *PDRS Console Handbook, Volume II*, JSC-17525, Johnson Space Center, Houston, TX: NASA.

JSC (June, 1990), *Orbiter Emergency/Caution and Warning Alarm Workbook*, C&W 2102, TD402, Johnson Space Center, Houston, TX: NASA.

Kasik, D.J., M.A. Lund, and H.W. Ramsey (January, 1989), "Reflections on Using a UIMS for Complex Applications", *IEEE Software*, pp. 54-61.

Kelly, Christine (1991), discussion about OMA Prototypes, Houston, TX: The MITRE Corporation.

Kessel, Karen (February, 1990), *Methodology and Communication Protocol for Graphical Interfaces to Intelligent Systems (GIIS)*, draft, Mountain View, CA: IntelliCorp.

Kuipers, B.M. and J.P. Kassirer (1984), "Causal Reasoning in Medicine: Analysis of a Protocol", *Cognitive Science*, No. 8, pp. 363-385.

Kuipers, B. (September, 1986), "Qualitative Simulation", *Artificial Intelligence*, No. 29.

Langlotz, C. P. and E. H. Shortliffe (1983), "Adapting a Consultation System to Critique User Plans", *International Journal of Man-Machine Studies*, 19:479--496.

LeClair, S., F. Abrams, and R. Matejka (1989), "Qualitative Process Automation: Self-directed Manufacture of Composite Materials", *AI EDAM*, 3(2), pp. 125-136.

Lemke, Andreas C. and Gerhard Fischer (August, 1990), "A Cooperative Problem Solving System for User Interface Design", *Proceedings of Eighth National Conference on Artificial Intelligence*, Volume 1, American Association for Artificial Intelligence, pp. 479-484.

Leveson, Nancy G. (February, 1991), "Software Safety in Embedded Computer Systems", *Communications of the ACM*, Volume 34, Number 2, pp. 34-46.

Lochbaum, Karen E. and Candace L. Sidner (August, 1990), "Models of Plans to Support Communication: An Initial Report", *Proceedings of Eighth National Conference on Artificial Intelligence*, Volume 1, American Association for Artificial Intelligence, pp. 485-490.

Malin, Jane T. and Debra L. Schreckenghost (April, 1990), "A Space Operations Framework for Computer-Human Interaction in Intelligent Computer-Aided Problem Solving Systems", position paper for Human Factors in Computing Systems Conference, CHI '90 Workshop: Computer-Human Interaction in Aerospace Systems.

Malin, Jane T. and Debra L. Schreckenghost (April, 1991), "Human-Computer Interaction Design is not just User Interface Design, position paper for Human Factors in Computing Systems Conference, CHI '91 Workshop: Advances in Computer-Human Interaction in Complex Systems.

Malin, J. T., D. L. Schreckenghost, and C. G. Thronesbery (April, 1991), "Human-Computer Interaction Must be Embedded in System Design: Lessons from NASA Intelligent Systems", poster session at Human Factors in Computing Systems Conference, CHI '91.

Malin, J. T., D. L. Schreckenghost, D. D. Woods, S. S. Potter, L. Johannesen, and M. Holloway (September, 1991), *Making Intelligent Systems Team Players: Case Studies and Design Issues, Volume 2. Fault Management System Cases*, Houston, TX: NASA - Johnson Space Center.

Mavrovouniotis, M. and G. Stephanopolous (July, 1987), "Reasoning with Orders of Magnitude and Approximate Relations", *Proceedings of National Conference on Artificial Intelligence*.

MDSSC (November, 1989), *Cooperating Expert System Lessons Learned*, Draft, Houston, TX: McDonnell Douglas Space Systems Company.

Miller, P. L. (1986), *Expert Critiquing Systems: Practice-Based Medical Consultation by Computer*, New York: Springer-Verlag.

Mitchell, Christine M. and R.A. Miller (May/June, 1986), "A Discrete Control Model of Operator Function: A Methodology for Information Display Design", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-16, Number 3, pp. 343-357.

Mitchell, Christine M. (July/August, 1987), "GT-MSOCC: A Domain for Research on Human-Computer Interaction and Decision Aiding in Supervisory Control Systems", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-17, Number 4, pp. 553-572.

Mitchell C. and D. Saisi (1987), "Use of Model-based Qualitative Icons and Adaptive Windows in Workstations for Supervisory Control Systems", *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17:573--593.

Mitchell, Christine M. (April, 1990), discussion at Human Factors in Computing Systems Conference, CHI '90 Workshop: Computer-Human Interaction in Aerospace Systems, Atlanta, GA: Georgia Institute of Technology.

Mitchell, Christine M. (April, 1991), discussion at Human Factors in Computing Systems Conference, CHI '91 Workshop: Advances in Computer-Human Interaction in Complex Systems, Atlanta, GA: Georgia Institute of Technology.

Miyata, Y. and D. A. Norman (1986), "Psychological Issues in Support of Multiple Activities", D. A. Norman and S. W. Draper, editors, *User Centered System Design: New Perspectives on Human-Computer Interaction*, Hillsdale, NJ: Lawrence Erlbaum.

Moray, N. (1986) "Modelling Cognitive Activities: Human Limitations in Relation to Computer Aids", E. Hollnagel, G. Mancini, and D. D. Woods, editors, *Intelligent Decision Support in Process Environments*, New York: Springer-Verlag.

Muir, B. (1987), "Trust between Humans and Machines", *International Journal of Man-Machine Studies*, 27:527--539.

Muratore, J.F., T.A. Heindel, T.B. Murphy, A.N. Rasmussen, and R.Z. McFarland (December, 1990), "Real-Time Data Acquisition at Mission Control", *Communications of the ACM*, Volume 33, Number 12, pp 18 - 31.

Norman, Donald (December, 1983), "Design Principles for Human-Computer Interfaces", *Proceedings of Human Factors in Computing Systems CHI '83*.

Norman, Don (April, 1990), discussion at Human Factors in Computing Systems Conference, CHI '90 Workshop: Computer-Human Interaction in Aerospace Systems, San Diego, CA: University of California.

Post, Stephen, and Andrew P. Sage (January/February, 1990), "An Overview of Automated Reasoning", *IEEE Transactions on Systems, Man, and Cybernetics*, Volume 20, Number 1, pp. 202-224.

Raiman, O. (August, 1986), "Order of Magnitude Reasoning", *Proceedings of National Conference on Artificial Intelligence*.

Rasmussen, A.N., J.F. Muratore, T.A. Heindel (August, 1990), "The INCO Expert System Project: CLIPS in Shuttle Mission Control", *Proceedings of First CLIPS Conference*, NCP 10049, Volume 1, Johnson Space Center, Houston, TX: NASA.

Rasmussen, J.(1986), *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*, New York: North-Holland.

Remington, Roger (April, 1990), discussion at Human Factors in Computing Systems Conference, CHI '90 Workshop: Computer-Human Interaction in Aerospace Systems, Moffett Field, CA: NASA-Ames Research Center.

Roberts, K. H. and D. M. Rousseau (1989), "Research in Nearly Failure-free, High Reliability Systems: 'Having the Bubble.'", *IEEE Transactions*, 36: 132-139.

Robertson, S. P., W. Zachary, and J. Black. [Eds] (1990), *Cognition, Computing and Cooperation*, New Jersey: Ablex.

Rochlin, G. I., T. R. La Porte, and K. H. Roberts (Autumn, 1987), "The Self-designing High-reliability Organization: Aircraft Carrier Flight Operations at Sea", *Naval War College Review*, 77-91.

Roth, E. M., K. Bennett, and D. D. Woods (1987), "Human Interaction with an 'Intelligent' Machine", *International Journal of Man-Machine Studies*, 27:479--525.

Roth, E. M. and D. D. Woods (1988), "Aiding Human Performance: I. Cognitive Analysis", *Le Travail Humain*, 51(1):39--64.

Roth, E. M. and D. D. Woods (1989), "Cognitive Task Analysis: An Approach to Knowledge Acquisition for Intelligent System Design", G. Guida and C. Tasso, editors, *Topics in Expert System Design*, New York: North-Holland.

Rouse, W. B., N. Geddes and R. E. Curry (1987-1988), "An Architecture for Intelligent Interfaces: Outline of an Approach to Supporting Operators of Complex Systems", *Human-Computer Interaction*, 3:87--122.

Rouse, W.B., N.D. Geddes, and J.M. Hammer (March, 1990), "Computer-aided Fighter Pilots", *IEEE Spectrum*, pp. 38-41.

Rudisill, Marianne (March 28, 1990), "Crew Display Standards, Definition, and Development", briefing to OMS Working Group, Houston, TX: Johnson Space Center.

Rudisill, Marianne (April, 1990), discussion at Human Factors in Computing Systems Conference, CHI '90 Workshop: Computer-Human Interaction in Aerospace Systems, Houston, TX: NASA-Johnson Space Center.

Schreckenghost, Debra L. (July, 1990) "PDRS Intelligent System Human-Computer Interface Design Concepts", briefing at Johnson Space Center, Houston TX: The MITRE Corporation.

Shafto, Michael G. and Roger W. Remington (April, 1990), "Building Human Interfaces to Fault Diagnostic Expert Systems II: Interface Development for a Complex, Real-Time System", Human Factors in Computing Systems Conference, CHI '90 Workshop: Computer-Human Interaction in Aerospace Systems, Moffett Field, CA: NASA - Ames Research Center.

Shalin, Valerie (April, 1990), discussion at Human Factors in Computing Systems Conference, CHI '90 Workshop: Computer-Human Interaction in Aerospace Systems, Minneapolis, Minnesota: Honeywell Systems and Research Center.

Shaw, John A. (March, 1988), "Reducing Operator Error in Distributed Control Systems", *InTech*, pp. 45-50.

Shepard, Terry (April, 1990), discussion at Human Factors in Computing Systems Conference, CHI '90 Workshop: Computer-Human Interaction in Aerospace Systems, Kingston, Ontario, Canada: Royal Military College of Canada.

Simmons, Reid and Randall Davis (August, 1987), "Generate, Test and Debug: Combining Associational Rules and Causal Models", *Proceedings of Tenth International Joint Conference on Artificial Intelligence*, Volume 2, pp. 1071-1078.

Smith, S.L. and J.N. Mosier (August, 1986), *Guidelines for Designing User Interface Software*, MTR-10090, Houston, TX: The MITRE Corporation.

Sorkin, R. D. and D. D. Woods (1985), "Systems with Human Monitors: A Signal Detection Analysis", *Human-Computer Interaction*, 1:49--75.

Suchman, L. A. (1987), *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge, England: Cambridge University Press.

van Creveld, M. (1985), *Command in War*, Cambridge, MA: Harvard University Press.

Weld, D. and J. de Kleer (1990), *Readings in Qualitative Reasoning about Physical Systems*, Los Altos, CA: Morgan Kaufmann.

Wexelblat, Richard L. (Fall, 1989), "On Interface Requirements for Expert Systems", *The AI Magazine*, Volume 10, Number 3, pp. 66-78.

Wick, M. R. and W. B. Thompson (1989), "Reconstructive Explanation: Explanation as Complex Problem Solving", *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*.

Wick, Michael R., (Fall, 1989), "The 1988 AAAI Workshop on Explanation", *The AI Magazine*, Volume 10, Number 3, pp. 22-26.

Wiener, E. L. (1989), *Human Factors of Advanced Technology ('Glass Cockpit') Transport Aircraft*, Technical Report 117528, NASA.

Wilford, Charlotte (1990), discussion about Space Station User Support Environment, formerly from Houston, TX: The MITRE Corporation.

Williams, B. (1984), "Qualitative Analysis of MOS Circuits", *Artificial Intelligence*, No. 24.

Woods, D.D. (1984), "Visual Momentum: A Concept to Improve the Cognitive Coupling of Person and Computer", *International Journal of Man-Machine Studies*, 21:229--244.

Woods, D. D. (1986), "Cognitive Technologies: The Design of Joint Human-machine Cognitive Systems", *AI Magazine*, 6:86--92.

Woods, D. D., W. C. Elm, and J. R. Easter (1986), "The Disturbance Board Concept for Intelligent Support of Fault Management Tasks", *Proceedings of the International Topical Meeting on Advances in Human Factors in Nuclear Power*, American Nuclear Society/European Nuclear Society.

Woods, D. D. (1988), "Coping with Complexity: The Psychology of Human Behavior in Complex Systems", L. P. Goodstein, H. B. Andersen, and S. E. Olsen, editors, *Mental Models, Tasks and Errors*, London: Taylor & Francis.

Woods, D. D. and E. M. Roth (1988), "Aiding Human Performance: II. From Cognitive Analysis to Support Systems", *Le Travail Humain*, 51(2):139--171.

Woods, D. D. and E. M. Roth (1988), "Cognitive Systems Engineering", M. Helander, editor, *Handbook of Human-Computer Interaction*, New York: North-Holland.

Woods, D. D. and G. Elias (1988), "Significance Messages: An Integral Display Concept", *Proceedings of the Human Factors Society 32nd Annual Meeting*.

Woods, D.D. and M. C. Eastman (1989), "Integrating Principles for Human-Computer Interaction into the Design Process, *IEEE International Conference on Systems, Man, and Cybernetics*.

Woods, D. D., E. M. Roth, and K. B. Bennett (1990), "Explorations in Joint Human-machine Cognitive Systems", S. Robertson, W. Zachary, and J. Black, editors, *Cognition, Computing and Cooperation*, Norwood, NJ: Ablex Publishing.

Woods, D.D., E. M. Roth, W. F. Stubler, and R. J. Mumaw (1990), "Navigating Through Large Display Networks in Dynamic Control Applications", *Proceedings of the Human Factors Society 34th Annual Meeting*.

Woods, D. D., L. Johannesen and S. S. Potter (September, 1990), *Structured Bibliography on Human Interaction with Intelligent Systems*, Johnson Space Center, Houston, TX: NASA.

Woods, D. D., J. T. Malin, and D. L. Schreckenghost (November, 1990), "Human-Computer Interaction with Intelligent Systems", briefings presented at seminar held at Marshall Space Flight Center, Huntsville, AL: NASA.

Woods, D. D. (1991), "The Cognitive Engineering of Problem Representations", G. R. S. Weir and J. L. Alty, editors, *Human-Computer Interaction and Complex Systems*, London: Academic Press, in press.

Woods, D.D., S.S. Potter, L. Johannesen, and M. Holloway (March, 1991), *Human Interaction with Intelligent Systems: Volume I -- Trends, Problems, New Directions*, CSEL Report 1991-001, Cognitive Systems Engineering Laboratory, Columbus, OH: Ohio State University.

Woolf, B., D. Blegen, J. Jansen, and A. Verloop (1986), "Teaching a Complex Industrial Process", *Proceedings of National Conference on Artificial Intelligence*.

Glossary

ADS	Advanced Document System
AI	Artificial Intelligence
ARC	Ames Research Center
BITE	Built-In Test Equipment
C&W	Caution and Warning
CAD	Computer-Aided Design
CAP	Crew Activity Plan
CIL	Critical Items List
COA	Course of Action
CODE	COMP Development Environment
CPU	Central Processing Unit
DAI	Distributed Artificial Intelligence
DESSY	Decision Support System
DMS	Data Management System
EAFB	Edwards Air Force Base
ECS	Environmental Control System
ED	Engineering Directorate
EECOM	Electrical, Environmental, Consumables, and Mechanical Systems
EVA	Extra-Vehicular Activity
FDF	Flight Data File
FDIR	Fault Detection, Isolation, and Recovery
FMEA	Failure Mode and Effects Analysis
GNC	Guidance, Navigation and Control
GPC	General Purpose Computers
HITEX	Human Interface to Thermal Expert System
HCI	Human-Computer Interaction
IESP	INCO Expert System Project
IS	Intelligent System
ILDSS	Intelligent Launch Decision Support System
INCO	Instrumentation and Communications Officer
ISB	Intelligent Systems Branch
ISD	Information Systems Directorate
ISE	Integrated Station Executive
ITF	Integrated Test Facility
JPL	Jet Propulsion Lab
JSC	Johnson Space Center
KATE	Knowledge-based Autonomous Test Engineer
KB	KiloByte
KSC	John F. Kennedy Space Center

LAN	Local Area Network
LESC	Lockheed Engineering and Science Corp.
LOS	Loss of Signal
MCC	Mission Control Center
MCCU	Mission Control Center Upgrade
MDSSC	McDonnell Douglas Space Systems Company
MED	Manual Entry Device
MER	Mission Evaluation Room
MOCR	Mission Operations Control Room
MP	Monitored Process
MPM	Manipulator Positioning Mechanism
MPSR	Multi-Purpose Support Room
MRL	Manipulator Retention Latch
MSFC	Marshall Space Flight Center
MSID	Measurement Stimulus Identification
NASA	National Aeronautics and Space Administration
NTD	NASA Test Director
OAET	Office of Aeronautics, Exploration, and Technology
OIS	Open Information Systems
OMA	Operations Management Application
OMS	Operations Management System
ONAV	Onboard Navigation
PBI	Push Button Indicator
PDRS	Payload Deployment and Retrieval System
PSI	Pounds per Square Inch
RAVES	Remotely Augmented Vehicle Expert Systems
RCS	Reaction Control System
REX	Rendezvous Expert System
RMS	Remote Manipulator System
RSOC	Rockwell Shuttle Operations Company
RTDS	Real-Time Data System
RTIDE	Real-Time Interactive Display Environment
RTIMES	Real-Time Interactive Monitoring Systems
RTOP	Research and Technology Operating Plan
SES	Space Shuttle Engineering Simulation
SELMON	Selective Monitoring System
SHARP	Spacecraft Health Automated Reasoning Prototype
SPAN	Spacecraft Analysis
SSCC	Space Station Control Center
SSM/PMAD	Space Station Module/Power Management and Distribution
STOL	Short Take Off and Landing
STRBD	Starboard
STS	Space Transportation System
TCS	Thermal Control System
TEXSYS	Thermal Expert System
TDRSS	Tracking and Data Relay Satellite System

UISIT
USE
VDU

User-Intelligent System Interface Toolkit
User Support Environment
Video Display Unit

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 1991		3. REPORT TYPE AND DATES COVERED Technical Memorandum	
4. TITLE AND SUBTITLE Making Intelligent Systems Team Players: Case Studies and Design Issues Volume 1: Human-Computer Interaction Design				5. FUNDING NUMBERS	
6. AUTHOR(S) Jane T. Malin*, Debra L. Schreckenghost**, David D. Woods***, Leila Johannesen***, Scott S. Potter***, Matthew Holloway***, Kenneth D. Forbus****					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Automation and Robotics Division NASA Johnson Space Center Houston, Texas 77058				8. PERFORMING ORGANIZATION REPORT NUMBER S-643	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, D.C. 20546-001				10. SPONSORING / MONITORING AGENCY REPORT NUMBER NASA TM 104738	
11. SUPPLEMENTARY NOTES *NASA **The MITRE Corporation, Houston, Texas ***Cognitive Systems Engineering Laboratory, Ohio State University ****The Institute for the Learning Sciences, Northwestern University					
12a. DISTRIBUTION / AVAILABILITY STATEMENT Unclassified/Unlimited Subject Category 59				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) Initial results are reported from a multi-year, interdisciplinary effort to provide guidance and assistance for designers of intelligent systems and their user interfaces. The objective is to achieve more effective human-computer interaction (HCI) for systems with real-time fault management capabilities (i.e., process monitoring and control during anomalous situations). Intelligent fault management systems within NASA have been evaluated for insight into the design of systems with complex HCI. Preliminary results include (1) a description of real-time fault management in aerospace domains, (2) recommendations and examples for improving intelligent system design and user interface design, (3) identification of issues requiring further research, and (4) recommendations for a development methodology integrating HCI design into intelligent system design.					
14. SUBJECT TERMS Human-computer interaction, user interface, intelligent system, design guidance, development methodology, real-time fault management				15. NUMBER OF PAGES	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL		